Contents of Appendices

A. Formal Statement of Iterative Magnitude Pruning

Algorithm 1 Iterative Magnitude Pruning (IMP) with weight rewinding to epoch 10 and N iterations.

```
1: Create a neural network with randomly initialized weights $W_0 \in \mathbb{R}^d$ and initial pruning mask $m = 1^d$
2: Train $W_0$ to epoch 10, resulting in weights $W_{10}$
3: for $n \in \{1, \ldots, N\}$ do
4:     Train $m \odot W_{10}$ (the element-wise product of $m$ and $W_{10}$) to final epoch $T$ and weights $m \odot W_{T,n}$
5:     Prune the 20% of weights in $m \odot W_{T,n}$ with the lowest magnitudes. $m[i] = 0$ if $W_{T,n}[i]$ is pruned
6: Return $m$ and $W_{T,n}$
```

B. Experimental Details

B.1. ResNets

We study the residual networks (ResNets) designed by He et al. (2016) for CIFAR-10 and ImageNet. ResNets for CIFAR-10 are composed of an initial convolutional layer, three sets of B residual blocks (each with two convolutional layers and a skip connection), and a linear output layer. The sets of blocks have 16, 32, and 64 convolutional channels, respectively.

ResNets for ImageNet and TinyImageNet are composed of an initial convolutional layer, a max-pooling layer, four sets of residual blocks (each with three convolutional layers and a skip connection), and a linear output layer. The sets of blocks have 64, 128, 256, and 512 convolutional channels, respectively. On ImageNet, we use a ResNet with 50 layers. On TinyImageNet, we use a ResNet with 18 layers. Both choices are standard for these datasets.

We place batch normalization before the ReLU activations.

To vary the width of the networks, we multiply the number of convolutional channels by the width scaling factor w. To vary the depth of the CIFAR-10 ResNets, we vary the value of B. The depth l of the network is the total number of the layers in the network, not counting skip connections.

B.2. VGG Networks

We study the VGG-16 variant of the VGG networks for CIFAR-10 as provided by the OpenLTH repository.[10] The network is divided into five sections, each of which is followed by max pooling with kernel size 2 and stride 2. The sections contain 3x3 convolutional layers arranged as follows:

Section	Width	Layers
1	64	2
2	128	2
3	256	3
4	512	3
5	512	3

The network has ReLU activations and batch normalization before each activation. To vary the width of VGG-16, we multiply each of the per-segment widths by the width scaling factor w.

B.3. DenseNets

We study the densely connected residual networks (DenseNets) designed by He et al. (2016) for CIFAR-10. DenseNets for CIFAR-10 are composed of an initial convolutional layer, four sets of dense blocks, and a linear output layer. Between the sets of blocks are transition layers of 1x1 convolutions and an average pooling operation that downsamples the image by 2x. Each block comprises a 1x1 convolution that increases the channel count by 4x and a 3x3 block that decreases it to a fixed constant size g; this output is then concatenated to the input of the block. As such, if the input to the block has n channels, the output of the block has $n + g$ channels. We use DenseNet-121, which has sets of 6, 12, 24, and 16 blocks. g is set to 16 but is multiplied by the width scaling factor w to modify the width.

B.4. Training Hyperparameters

We train CIFAR-10 and SVHN ResNets and VGG-16 for 160 epochs with a batch size of 128. The initial learning rate is 0.1, and it drops by an order of magnitude at epochs 80 and 120. We optimize using SGD with momentum (0.9). We initialize with He uniform initialization. CIFAR-10 data is augmented by normalizing, randomly flipping left and right, and randomly shifting by up to four pixels in any direction (and cropping afterwards). SVHN data is not augmented.

We train CIFAR-10 DenseNets with the same hyperparameters but for 200 epochs (with learning rate drops at 130 and 165 epochs). We train SVHN DenseNets with the same hyperparameters but for 100 epochs (with learning rate drops at 70 and 85 epochs).

[10] `github.com/facebookresearch/open_lth`

We train ImageNet ResNets for 90 epochs with a batch size of 1024. The initial learning rate is 0.4, and it drops by an order of magnitude at epochs 30, 60, and 80. We perform linear learning rate warmup from 0 to 0.4 over the first 5 epochs. We optimize using SGD with momentum (0.9). We initialize with He uniform initialization. Data is augmented by normalizing, randomly flipping left and right, selecting a random aspect ratio between 0.8 and 1.25, selecting a random scaling factor between 0.1 and 1.0, and cropping accordingly.

We train TinyImageNet ResNets identically except for that we train them for 200 epochs (with learning rate drops at 100 and 150 epochs) and a learning rate of 0.2. Augmentation is identical to ImageNet.

B.5. Dimensions

We use the following dimensions for the additional experiments. We select configurations using the same methodology as in Table 1.

Network Family	Densities (d)	Depths (l)	Width Scalings (w)	Subsample Sizes (n)
CIFAR-10/SVHN ResNet	$0.8^i, i \subseteq \{0, \ldots, 40\}$	$l \subseteq$ {8, 14, 20, 26, 50, 98}	$2^i, i \subseteq \{-4, \ldots, 2\}$	$\frac{N}{i}, i \in \{1, 2, 4, 8, 16, 32, 64\}$
CIFAR-10/SVHN VGG	$0.8^i, i \subseteq \{0, \ldots, 37\}$	16	$2^i, i \subseteq \{-4, \ldots, 0\}$	$\frac{N}{i}, i \in \{1\}$
CIFAR-10/SVHN DenseNet	$0.8^i, i \subseteq \{0, \ldots, 50\}$	121	$2^i, i \subseteq \{-4, \ldots, 1\}$	$\frac{N}{i}, i \in \{1\}$
TinyImageNet ResNet	$0.8^i, i \subseteq \{0, \ldots, 50\}$	18	$2^i, i \subseteq \{-6, \ldots, 0\}$	$\frac{N}{i}, i \in \{1\}$
ImageNet ResNet	$0.8^i, i \subseteq \{0, \ldots, 30\}$	50	$2^i, i \subseteq \{-4, \ldots, 0\}$	$\frac{N}{i}, i \in \{1, 2, 4\}$

C. Full Data for Key Observations in Section 3

In this appendix, we show that our observations from Section 3 hold when varying all dimensions (depth, width, and dataset size) on both the CIFAR-10 and ImageNet ResNets for IMP. Figure 9 shows the error versus density when changing width (left) depth (center) and data (right). In Figure 10, we similarly show the dependency of the error on density for Imagenet when varying width (left) and dataset size (right).

In Figure 9, we observe that all curves have a similar slope in the power-law region. In Equation 1, this implies that while γ is allowed to vary with l, w and n, it is in practice approximately a constant. Similarly, the high-error plateau $\epsilon^{\uparrow}$ is also shared across curves such that it too is approximately constant. In contrast, the transition from high-error plateau to the power-law region is not constant as a function of density. Section 4 finds exactly this dependency of the transition parameter p.

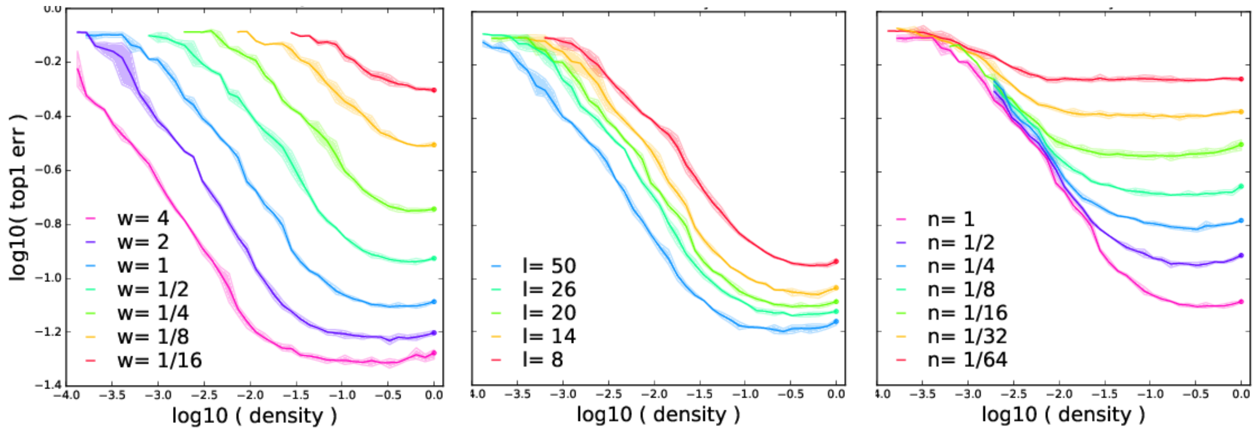

Figure 9. Relationship between density and error when pruning CIFAR-10 ResNets and varying w (left, $l = 20$, $n = N$), l (center, $w = 1$, $n = N$), n (right, $l = 20$, $w = 1$)

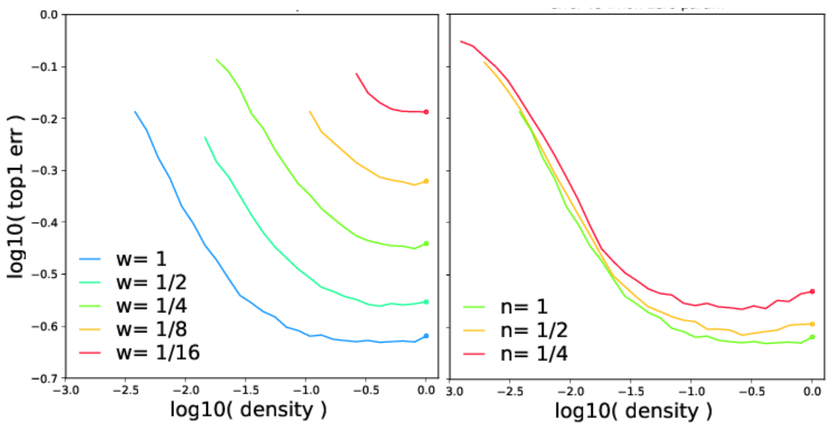

Figure 10. Relationship between density and error when pruning Imagenet ResNet-50 and varying w (left, $n = N$), and n (right, $w = 1$)

D. Partial (Projections) Fit Results for Section 4

In Section 4, we fit the error jointly as a function of all dimensions showing that Equation 2 provides a good approximation to the error in practice. In this appendix, we consider important sub-cases, such as the case when one wishes to scale only one degree of freedom while pruning. This serves both a practical scenario, but also allows for a qualitative visualization of the fit (and typical sources of error), which is otherwise difficult to perform over all dimensions jointly. From a practical standpoint, in this case one need not estimate the parameters associated with the fixed degree of freedom.

Recall that, given the non-pruned network error ϵ_{np}, all dependencies on the individual structural degrees of freedom l, w are captured by the invariant $m^* \triangleq l^\phi w^\psi d$. This means that, if one wishes to estimate the error while pruning when holding width fixed, we need not estimate ψ. Similarly if depth is held constant, we need not estimate ϕ.

Figure 11 shows these partial fits. Shown from left to right are the fits done while pruning and varying width, depth and data respectively. Correspondingly, these fits omit separately ψ or ϕ or omit both when depth nor width are scaled. The fits were performed with all available density points for each dimension. For CIFAR-10: 7 widths, 224 points for the width partial fit; 7 dataset fractions, 240 points for the data partial fit; 4 depths, 164 points for the depth partial fit. For ImageNet: 5 widths, 83 points for the width partial fit; 3 dataset fractions, 86 points for the data partial fit.

This exercise, apart from its practical implications, highlights the fact that there are in effect two groups of parameters comprising the estimation. The first are the parameters $\epsilon^\uparrow$, γ and p' which control the dependency as a function of density (or more generally, as a function of the invariant). The second are ϕ and ψ which are properties of the architectural degrees of freedom captured by the invariant. Moreover, within the first group of parameters $\epsilon^\uparrow$, γ, can be isolated and found from a single pruning curve, as they are not a function of l, w, n.

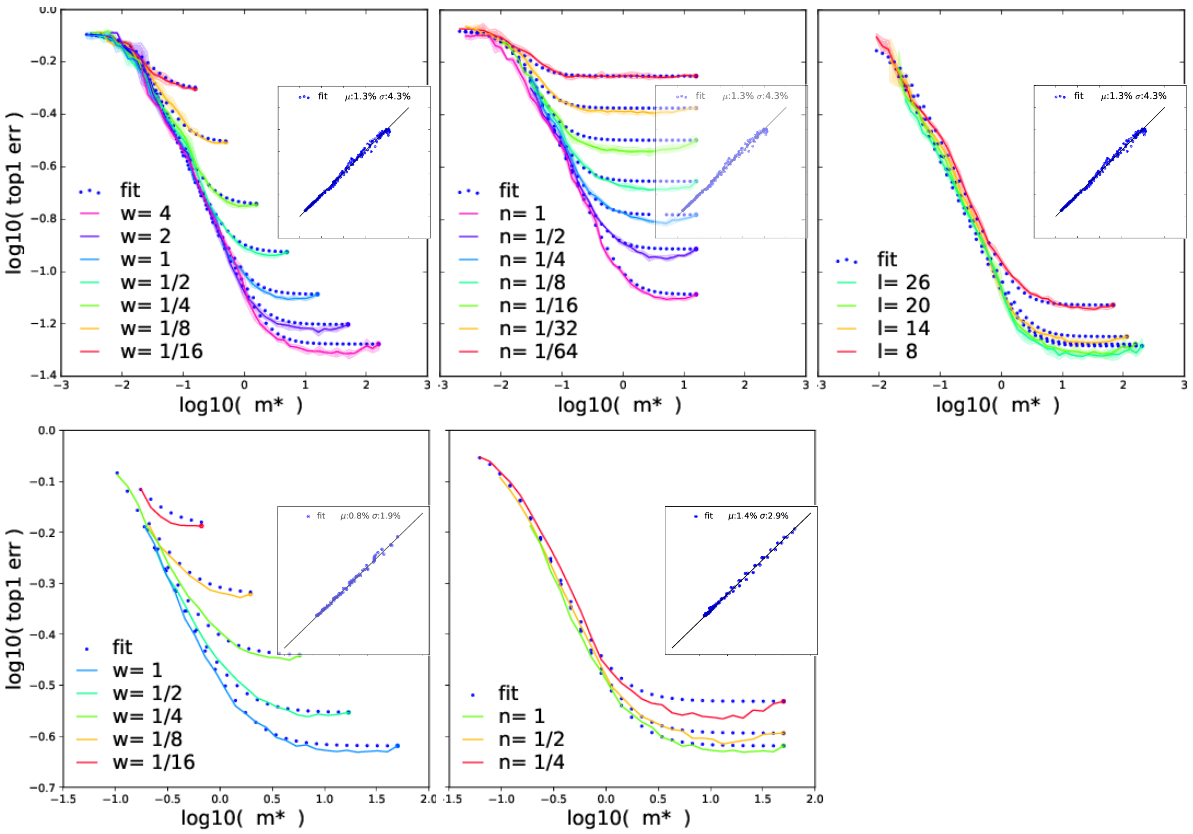

Figure 11. Top row: CIFAR-10. Bottom row: ImageNet. Left: varying width. Center: varying dataset size. Right: varying depth. Lines are the actual error and dots are the estimated error.

E. Additional Architectures and Datasets

In this appendix, we show that our functional form applies to additional pairs of networks and datasets: (CIFAR-10 ResNet, SVHN), (VGG-16, CIFAR-10), (VGG-16, SVHN), (DenseNet-121, CIFAR-10), (DenseNet-121, SVHN), (ImageNet ResNet-18, TinyImageNet).

In general, we obtain good fits on CIFAR-10 and TinyImageNet. On SVHN, fits are worse but the networks suffer from high measurement error (i.e., accuracy varies greatly between multiple runs at the same density) at low densities; nevertheless, fits are often better than measurement error because they average out some of the error.

We add these additional comparisons in the following Figures:

- Figure 12: ResNet-20 on SVHN with IMP as width varies. $|\mu| < 2\%$, $\sigma < 8\%$. Notably, the measurement error in this case is large ($\sigma \sim 9.5\%$), dominating (over the approximation error) the total fit error. The fit averages out some of this error, resulting in a fit error which is lower than the measurement error. In general, experiments on SVHN are quite noisy, leading to significant measurement error.
- Figure 13: VGG-16 on CIFAR-10 with IMP as width varies. $|\mu| < 3\%$, $\sigma < 7\%$ (compared to measurement error 12%).
- Figure 14: VGG-16 on SVHN with IMP as width varies. $|\mu| < 4\%$, $\sigma < 15\%$ (compared to measurement error 20%—measurement error is large at very low densities).
- Figure 15: DenseNet-121 on CIFAR-10 with IMP as width varies. $|\mu| < 1\%$, $\sigma < 8\%$. Bias is evident in the transition as we discuss in Section 6.
- Figure 16: DenseNet-121 on SVHN with IMP as width varies. $|\mu| < 4\%$, $\sigma < 19\%$ (compared to measurement error 16%—measurement error is large at very low densities).
- Figure 17: ResNet-18 on TinyImageNet with IMP as width varies. $|\mu| < 1\%$, $\sigma < 1.3\%$ (compared to measurement error 1%).

Note that SynFlow suffers from exploding activations on deeper networks, so we do not vary ResNet depth in any of our SynFlow experiments.

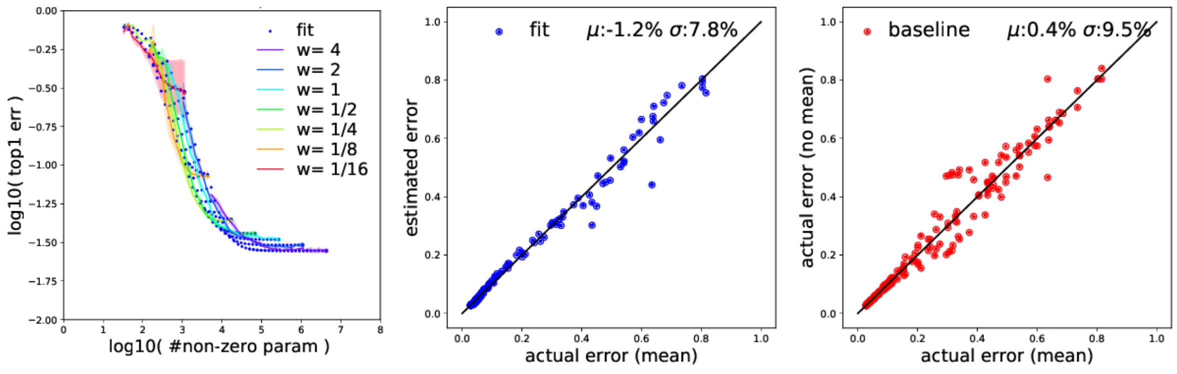

Figure 12. Fit for ResNet-20 on SVHN with IMP pruning.

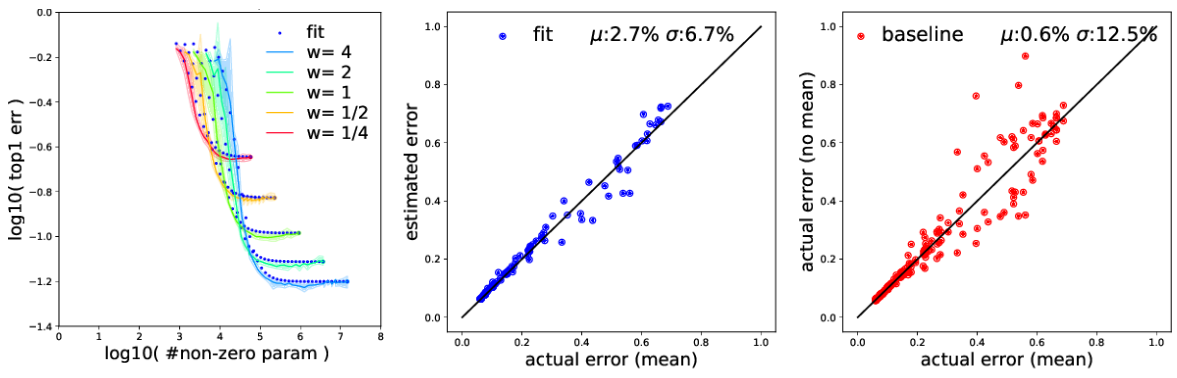

Figure 13. Fit for VGG-16 on CIFAR-10 with IMP pruning.

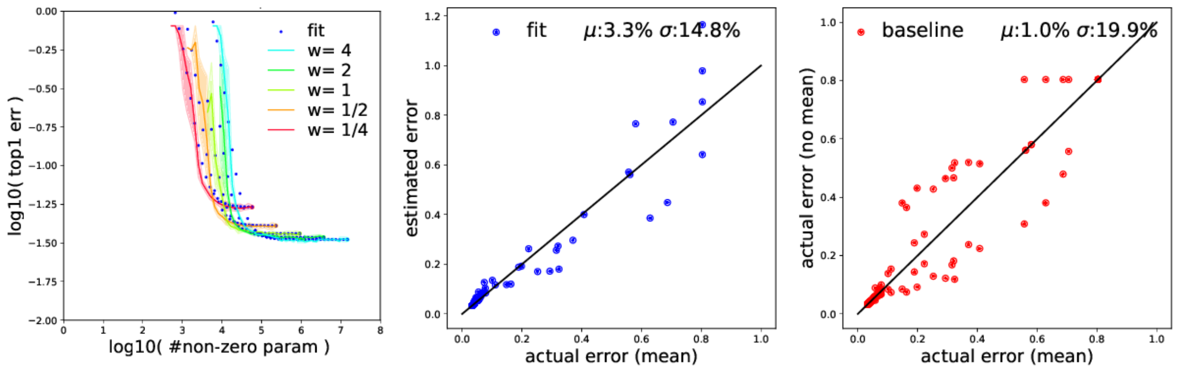

Figure 14. Fit for VGG-16 on SVHN with IMP pruning.

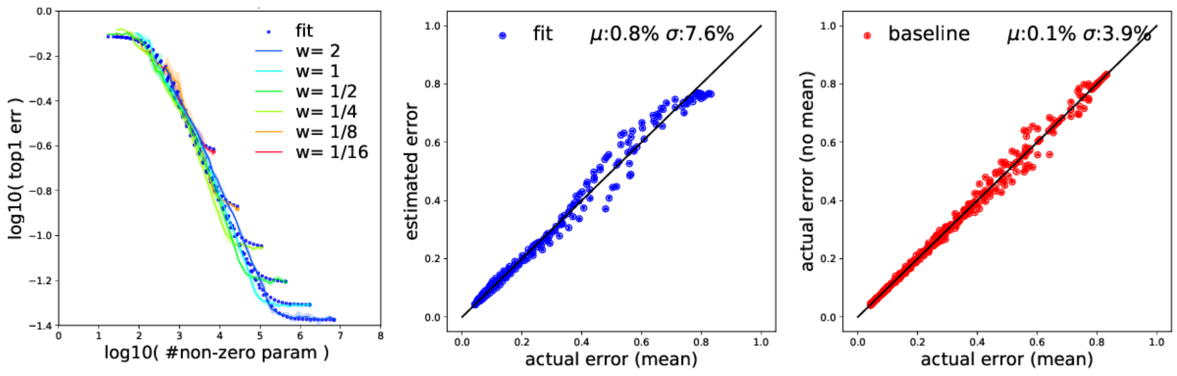

Figure 15. Fit for DenseNet-121 on CIFAR-10 with IMP pruning.

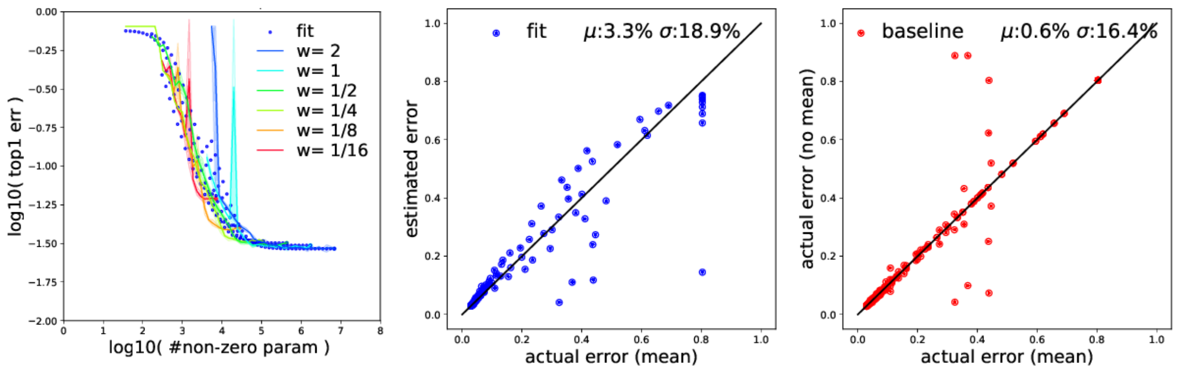

Figure 16. Fit for DenseNet-121 on SVHN with IMP pruning.

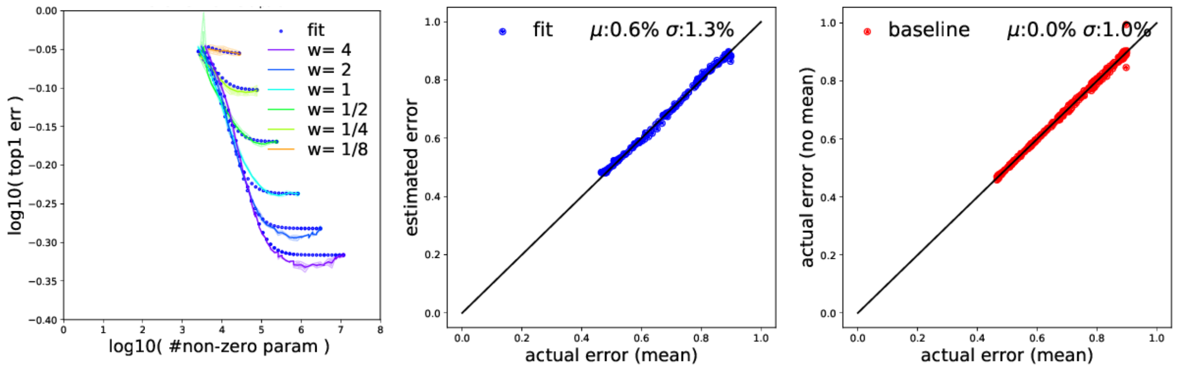

Figure 17. Fit for ResNet-18 on TinyImageNet with IMP pruning.

F. Towards Extrapolation

Background. In the main body, we showed that our scaling law accurately fits the error of pruned neural networks. As, such it has predictive power, allowing us to reason in a principled manner about pruning trade-offs. Similarly, it allows to make predictions about what would happen at larger model and data scales than explored here. Importantly, only a few experiments need be performed to find the coefficients for the scaling law (see Appendix 5).

However, we could ask, how accurately can we estimate the scaling law parameters from even smaller scales? That is, is it possible to fit our scaling law to data from networks with deliberately smaller depths, widths, and dataset sizes and accurately predict the error of larger-scale models? If so, we could make informed decisions about pruning large-scale models through small-scale experiments alone, saving the costs associated with large scale training and pruning.

Outside the context of pruning, the scaling laws of (Rosenfeld et al., 2020) (for both language models and image classification) and (Kaplan et al., 2020) (for predicting the expected performance of GPT-3 (Brown et al., 2020) at very large scale) have been shown to extrapolate successfully in this manner.

Results on CIFAR-10. In Figure 18, we show the result of extrapolating from small-scale networks on CIFAR-10 ($w = \frac{1}{8}, \frac{1}{4}$; $l = 14, 20$) to all widths and depths on CIFAR-10. Extrapolation prediction is still accurate: $\mu < 7\%$, $\sigma < 6\%$ (vs. $\mu < 1\%$, $\sigma < 6\%$ in the main body).

Future work. However, extrapolation is particularly sensitive to systemic errors. Specifically, the transitions and the error dips can lead to large deviations when extrapolating. For ImageNet, the error dips (especially on small dataset sizes) are especially pronounced, preventing stable extrapolation. In order to improve extrapolation performance, future work should explore the challenges we discuss in Section 6: approaches to either model or mitigate these dips and to improve the fit of the transitions.

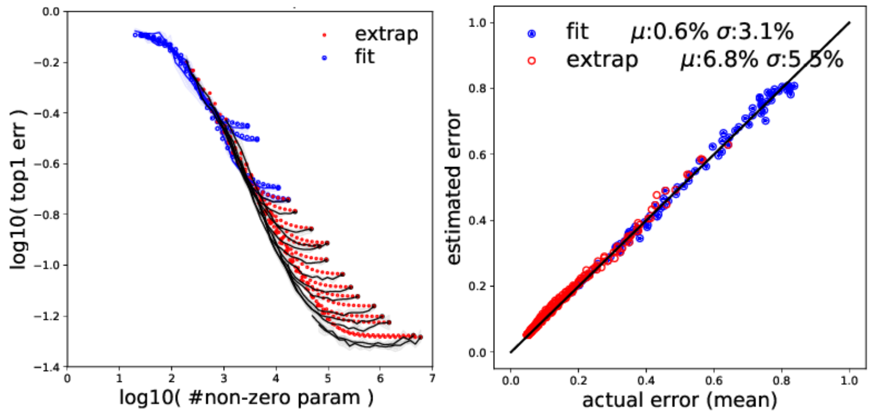

Figure 18. Extrapolation results from four pruned networks on CIFAR10 $w = \frac{1}{8}, \frac{1}{4}; l = 14, 20$ to all larger networks $(n = 1)$. Fit results are in blue, extrapolation in red, actual in black. Error versus number of non-zero parameters (left). Estimated versus actual errors (right).

G. Comparison of Pruning and Non-pruning Scaling Laws

In this appendix, we contrast the behavior of the error when pruning with the behavior of the error in the non-pruning setting. Hestness et al. (2017) show the the error follows a saturating power-law form when scaling data (with both low and high-error plateaus) but does not model them. Rosenfeld et al. (2020) unify the dependency on data and model size while approximating the transitions between regions; they propose the following form:

$$\tilde{\epsilon}(m,n) = an^{-\alpha} + bm^{-\beta} + c_\infty \tag{3}$$

$$\hat{\epsilon}(m,n) = \epsilon_0 \left\| \frac{\tilde{\epsilon}(m,n)}{\tilde{\epsilon}(m,n) - j\eta} \right\| \tag{4}$$

where m is the total number of parameters and n is the dataset size. $a, b, \alpha, \beta, c_\infty$, and η are constants, and ϵ_0 plays the role of $\epsilon^\uparrow$ in our notation.

Rosenfeld et al. model the upper transition—from power-law region to the high-error plateau—by a rational form in a fashion similar to the approach we take. The key difference is that we consider a power of the polynomials in the numerator and denominator of the rational form, where in Eq. 3 the power is hidden in the term $\tilde{\epsilon}$.

The biggest difference arises when considering the lower transition (between the low-error plateau and the power-law region). This transition is captured by Eq. 3. Considering either the width or depth degrees of freedom $x \in \{w, l\}$, Eq. 3 can be re-written as:

$$\tilde{\epsilon}(x) = b_x x^{-\beta_x} + c_x \tag{5}$$

Where b_x and β_x are constants and c_x is a constant as a function of x (it is only a function of the data size n).

Figure 19 (right) shows the error versus depth for different dataset sizes. In grey is the actual error, while in red is the best fit when approximating the error by Eq. 5. Qualitatively, one sees that the fit using Eq. 5 does indeed closely match the error in practice.

Recall that we are interested in comparing the errors as a function of the density. A requirement from any functional form used to model the dependency on the density is to degenerate to the error of the non pruned model ϵ_{np} at $d = 1$. We adapt Eq. 5 by solving the relation between b_x and c_x meeting this constraint, to arrive at:

$$\tilde{\epsilon}(x) = b_x x^{-\beta_x} + \epsilon_{np} - b_x \tag{6}$$

Contrast Eq. 5 with the functional form we propose in Eq. 1, re-written here for convenience:

$$\hat{\epsilon}(d, \epsilon_{np} \mid l, w, n) = \epsilon_{np} \left\| \frac{d - jp\left(\frac{\epsilon^\uparrow}{\epsilon_{np}}\right)^{\frac{1}{\gamma}}}{d - jp} \right\|^{\gamma} \text{ where } j = \sqrt{-1} \tag{7}$$

This can be simplified to capture only the lower transition—far enough from the upper transition ($d \gg p$)—to:

$$\hat{\epsilon}(d, \epsilon_{np} \mid l, w, n) = \epsilon_{np} \left\| \frac{d - jp\left(\frac{\epsilon^\uparrow}{\epsilon_{np}}\right)^{\frac{1}{\gamma}}}{d} \right\|^{\gamma} \tag{8}$$

Figure 19 (left) shows error versus density for different widths. In blue is the fit with Eq. 8 which follows closely the actual error (black) while in red is the fit with Eq. 6 which deviates noticeably in comparison.

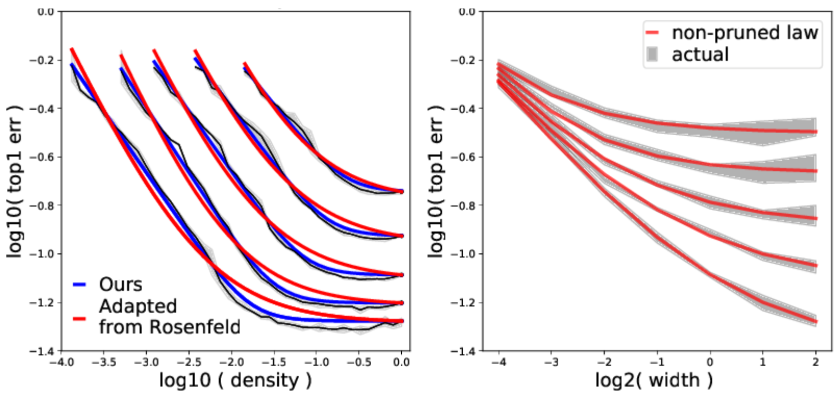

Figure 19. (Left) Error versus density for different widths. In blue is the fit eq. 2 follows closely the actual error (black) while in red is the fit for the adapted from Rosenfeld et al. (2020) which deviates noticeably in comparison. (Right) error of non-pruned networks versus width for different data, fit shown (solid red) for the non-pruning scaling from Rosenfeld et al. (2020).

We have seen that in practice that the form of Eq. 6 does not match well the pruning case, where the mismatch originates from lower transition shape. We have thus reached a phenomenological observation distinguishing the pruning and non-pruning forms; we leave the study of the origins of this phenomenon for future work.

H. The effect of error dips on estimation bias

In this appendix, we consider the effect of the error dips on our estimator as discussed in Section 4. As we mention in that section, when pruning a network, error often dips below ϵ_{np} during the low-error plateau.

Recall that we find the parameters in our estimator (Equation 2) by minimizing the MSE of relative error δ. Our estimation has bias if $\mathbb{E}\left(\hat{\epsilon} - \epsilon\right) \neq 0$ where the expectation is over all model and data configurations. Equivalently, the relative bias is $\mu \triangleq \mathbb{E}\delta = 0$ iff the estimator is unbiased. The Estimator captured by the joint form in Equation 2 is a monotonically increasing function of the density. It is also constrained such that at density $d = 1$ it is equal to the non-pruned error ϵ_{np}. It thus, can not reduce The MSE to zero, as it can not decrease to match the actual error dips. This results in the bias of the relative error μ which in practice is $\sim 1\%$.