On the Predictability of Pruning Across Scales

Jonathan S. Rosenfeld[1] **Jonathan Frankle**[1] **Michael Carbin**[1] **Nir Shavit**[12]
{jonsr,jfrankle,mcarbin,shanir}@csail.mit.edu
[1] Massachusetts Institute of Technology [2] Neural Magic Inc

Abstract

We show that the error of magnitude-pruned networks follows a scaling law, and that this law is of a fundamentally different nature than that of unpruned networks. We functionally approximate the error of the pruned networks, showing that it is predictable in terms of an invariant tying width, depth, and pruning level, such that networks of vastly different sparsities are freely interchangeable. We demonstrate the accuracy of this functional approximation over scales spanning orders of magnitude in depth, width, dataset size, and sparsity for CIFAR-10 and ImageNet. As neural networks become ever larger and more expensive to train, our findings enable a framework for reasoning conceptually and analytically about pruning.

1 Introduction

For decades, *pruning* has been a popular approach for reducing the size or computational demands of neural network inference [LeCun et al., 1990, Reed, 1993, Han et al., 2015]. Although the details differ between specific pruning techniques, most share the same high-level structure: (1) train the network for some amount of time, (2) delete unwanted parts of the network according to a heuristic (e.g., magnitude) to improve a desired characteristic at inference-time (e.g., smaller storage size or faster inference on specific hardware), and (3) train the network further to recover any accuracy lost when pruning [Han et al., 2015]. In practice, pruning reduces the parameter-counts of contemporary models by 2x [Gordon et al., 2020] to 5x [Renda et al., 2020] with no reduction in accuracy.

More than 80 pruning techniques have been published in the past decade [Blalock et al., 2020], each proposing a different instantiation of the aforementioned routine. Despite this enormous volume of research, there remains little guidance on important aspects of pruning. Consider a seemingly simple question one might ask when using a particular pruning technique:

Given a family of neural networks (e.g., ResNets on ImageNet of various widths and depths), which family member should we prune (and by how much) to obtain the network with the smallest parameter-count such that error does not exceed some threshold ϵ_k?

As a first try, we could attempt to answer this question using brute force: we could prune every member of a family (i.e., perform grid search over widths and depths) and select the smallest pruned network that satisfies our constraint on error. However, depending on the technique, pruning one network (let alone grid searching) could take days or weeks on expensive hardware.

If we want a more efficient alternative, we will need to make assumptions about pruned networks: namely, that there is some *structure* to the way that they behave. For example, that pruning a particular network changes the error in a predictable way. Or that changing the width or depth of a network changes the error when pruning it in a predictable way. We could then train a smaller number of networks, develop an idea of this structure, and estimate the answer to our question.

Preprint. Under review.

We have reason to believe that such structure does exist for pruning; techniques already take advantage of it implicitly. For example, Cai et al. [2019] create a single neural network architecture that can be scaled down to many different sizes; to choose which subnetwork to deploy, Cai et al. train an auxiliary, black-box neural network to predict subnetwork performance.

Structure has also been observed—and, even further, codified explicitly—for other aspects of deep learning. Tan and Le [2019] design the *EfficientNet* family by developing a heuristic for predicting efficient tradeoffs between depth, width, and resolution. Hestness et al. [2017] observe a power-law relationship between dataset size and the error of vision and NLP models. Rosenfeld et al. [2020] use a power law to predict the error of all variations of architecture families and dataset sizes jointly, for computer vision and natural language processing settings. Kaplan et al. [2020] develop a similar power law for language models that incorporates the computational cost of training.

Inspired by Rosenfeld et al. [2020] and Kaplan et al. [2020], we develop a scaling law to predict the error of neural networks as depth, width, and dataset size vary. Novel to our work, this scaling law predicts the error when *pruning* these networks. Our key insight is to treat the *density* of a pruned model as another architectural degree of freedom alongside width and depth.

We begin by developing a functional form that accurately estimates the generalization error of a specific model as it is pruned (Section 3). When pruning a network, error behaves in a materially different fashion than when varying its width or depth without pruning; we therefore choose a functional form that is materially different from those used in prior work. We expand this into a scaling law that jointly considers pruning alongside width, depth, and dataset size (Section 4). The basis for this scaling law is an *invariant* we uncover that describes ways that we can interchange depth, width, and pruning without affecting error. The result is a scaling law that accurately predicts the performance of pruned networks across scales. In summary, our contributions are as follows:

- We develop a scaling law that accurately estimates the error when pruning a single neural network with unstructured magnitude pruning.
- We observe and characterize an *invariant* that allows error-preserving interchangeability among depth, width, and pruning density.
- Using this invariant, we extend our single-network scaling law into a joint scaling law that predicts the error of all members of a network family at all dataset sizes and all pruning densities.
- In doing so, we demonstrate that there is structure to the behavior of the error of pruned networks that we can capture explicitly with a simple functional form.
- This functional form enables a framework in which we can reason conceptually and analytically about pruning.

This framework provides an elegant approach for answering our motivating question and other similar questions: with our functional form for pruning in hand, finding answers becomes an analytical exercise on an optimization problem.

2 Experimental Setup

Pruning. To prune neural networks, we use *iterative magnitude pruning* (IMP) [Janowsky, 1989, Han et al., 2015, Frankle et al., 2020]. IMP prunes by removing a fraction—typically 20%, as we do here—of individual weights with the lowest magnitudes at the end of training.[1] We choose these weights globally throughout the network, i.e., without regard to specific layers. We use per-weight magnitude pruning because it is generic, well-studied [Han et al., 2015], and matches the sparsity/accuracy tradeoffs of more complicated methods [Gale et al., 2019, Blalock et al., 2020, Renda et al., 2020].

Pruning weights typically reduces the accuracy of the trained network, so it is standard practice to further train after pruning to recover accuracy. We do so using a practice called *weight rewinding*, in which the values of unpruned weights are *rewound* to their values at epoch 10 and the training process is repeated from there to completion. To achieve sparsity levels beyond 20%, this process is repeated *iteratively*—pruning by 20%, rewinding, and retraining—until a desired sparsity level is reached. Renda et al. [2020] demonstrate that IMP with weight rewinding achieves state-of-the-art tradeoffs between sparsity and accuracy. For a formal statement of this pruning algorithm, see Appendix A.

[1] We do not prune biases or BatchNorm, so pruning 20% of weights prunes fewer than 20% of parameters.

Datasets. We study the image classification tasks CIFAR-10 and ImageNet. Our scaling law predicts the error when training with the entire dataset and smaller *subsamples*. To subsample a dataset to a size of n, we randomly select n of the training examples without regard to individual classes such that in expectation we preserve the original dataset distribution (we always retain the entire test set). When performing iterative pruning, we maintain the same subsample for all pruning iterations.

Networks. We study the residual networks (ResNets) designed by He et al. [2016] for CIFAR-10 and ImageNet (see Appendix B for full details on architectures and hyperparameters). We develop a scaling law that predicts the error (when pruned) of an entire *family* of ResNets with varying widths and depths. To vary width, we multiply the number of channels in each layer by a width scaling factor. To vary depth of the CIFAR-10 ResNets, we vary the number of residual blocks; we do not vary the depth of the ImageNet ResNets. We refer to a ResNet by its *depth* l (the total number of layers in the network, not counting skip connections) and its *width scaling factor* w.

Notation and terminology. Throughout the paper, we use the following notation:

- $\mathbb{D}_N = \{\boldsymbol{x}_i, y_i\}_{i=1}^N$ is a labeled training set with N examples. A *subsample* of size n is a subset of $\mathbb{D}_N$ with containing n examples selected uniformly at random.
- l and w are, respectively, the depth (i.e., the number of layers, excluding skip connections) and the width scaling factor of a particular network.
- A collection of networks that vary by width and depth are a *network family*.
- s is the *sparsity* of a pruned network (i.e., the fraction of weights that have been pruned) and $d \triangleq 1 - s$ is the *density* (i.e., the fraction of weights that have not been pruned).
- $\epsilon\left(d, l, w, n\right)$ is the test error of a network with the specified density, depth, and width scaling factor, and dataset size.
- $\epsilon_{np}\left(l, w, n\right) = \epsilon\left(1, l, w, n\right)$ is the test error of the unpruned network with the specified depth, width scaling factor, and dataset size. When clear from context, we omit (w, l, n) and write ϵ_{np}.
- $\hat{\epsilon}\left(\epsilon_{np}, d, l, w, n\right)$ is an estimate of the error for the specified unpruned error and d, l, w, and n.
- $\hat{\epsilon}(\epsilon_{np}, d \mid l, w, n)$ denotes the conditional form of our estimate. In contrast to $\hat{\epsilon}\left(\epsilon_{np}, d, l, w, n\right)$, it allows any parameters within the estimate's definition to be functions of w, l, and n.

Dimensions. In developing scaling laws, we vary four different dimensions in our experiments: dataset subsample size (n) and network degrees of freedom density (d), network depth (l), and width scaling factor (w). We consider the following ranges of these values in our experiments:

Network Family	N_{train}	N_{test}	Densities (d)	Depths (l)	Width Scalings (w)	Subsample Sizes (n)
CIFAR-10 ResNet	50K	10K	$0.8^i, i \in \{0, \ldots, 40\}$	8, 14, 20, 26, 50, 98	$2^i, i \in \{-4, \ldots, 2\}$	$\frac{N}{i}, i \in \{1, 2, 4, 8, 16, 32, 64\}$
ImageNet ResNet	1.28M	50K	$0.8^i, i \in \{0, \ldots, 30\}$	50	$2^i, i \in \{-4, \ldots, 0\}$	$\frac{N}{i}, i \in \{1, 2, 4\}$

3 Modeling the Error of a Pruned Network

Our goal in this section is to develop a functional form that accurately models the error of a member of a network family as it is pruned (using IMP with weight rewinding) based on its unpruned error $\epsilon_{np}(w, l, n)$. In other words, we wish to find a function $\hat{\epsilon}(\epsilon_{np}, d \mid l, w, n)$ that predicts the error at each density d for a network with a particular depth l, width scaling factor w, and dataset size n.

Key observations. Since IMP prunes a network 20% at a time, it produces pruned networks at intermediate levels of density $d_k = 0.8^k$ in the process of creating a final pruned network at density $d_K = 0.8^K$. In Figure 1 (left), we plot the error of these pruned networks for members of the CIFAR-10 ResNet family with a different widths w. All of these curves follow a similar pattern:[2]

Observation 1: Low-error plateau. The densest pruned networks (right part of the curves) have approximately the same error as the unpruned network: $\epsilon_{np}(w)$. We call this the *low-error plateau*.

Observation 2: Power-law region. When pruned further, error increases in a linear fashion on the logarithmic axes of the figure. Linear behavior on a logarithmic scale is the functional form of a *power law*, in which error relates to density through an exponent γ and a coefficient c: $\epsilon(d, w) \approx cd^{-\gamma}$. In particular, γ controls the slope of the line on the logarithmic axes.

[2]The same patterns occur when varying l and n for CIFAR-10 and w and n for ImageNet (Appendix C). We focus on varying width for CIFAR-10 here for illustrative purposes.

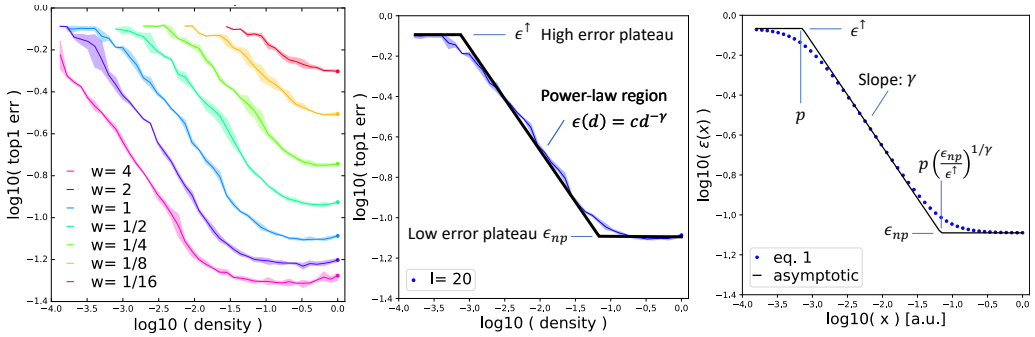

Figure 1: (left) Relationship between density and error when pruning CIFAR-10 ResNets; l varies, $w = 1$, $n = N$. (center) Low-error plateau, power-law region, and high-error plateau for CIFAR-10 ResNet $l = 20$, $w = 1$, $n = N$. (right) Visualizing Equation 1 and the roles of free parameters.

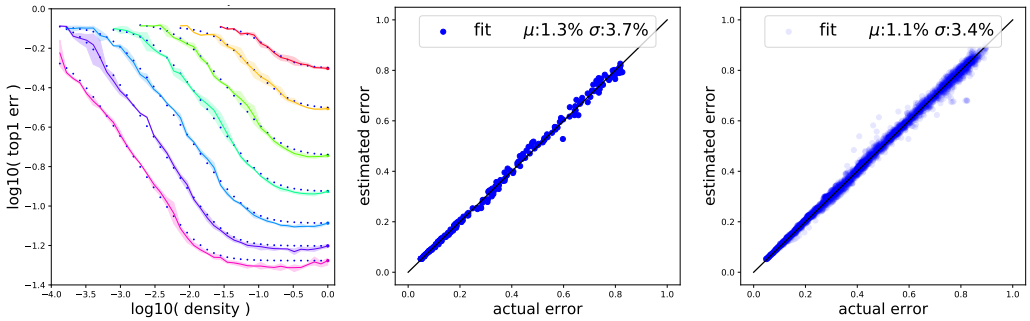

Figure 2: (left) Estimated (blue dots) and actual error (solid lines) when pruning CIFAR-10 ResNets; w varies, $l = 20$, $n = N$. (center) Estimated versus actual error for the networks in (left). (right) Estimated versus actual error for all CIFAR-10 ResNet configurations.

Observation 3: High-error plateau. When pruned further, error again flattens; we call this the *high-error plateau* and call the error of the plateau $\epsilon^\uparrow$.

Figure 1 (center) labels these regions for CIFAR-10 ResNet-20 (width scaling factor 1, dataset size N) and shows an approximation of these regions that is piece-wise linear on logarithmic axes. These observations are our starting point for developing a functional form that estimates error when pruning.

Functional form. Our next task is to find a functional form that accurately captures these observations about the relationship between density and error. In prior work, Rosenfeld et al. [2020] observe that the relationship between width and error shares the same general shape: it has a region of lower error, a power-law region, and region of higher error. However, this relationship is different enough from the one we observe (see Appendix D) to merit an entirely new functional form.

To develop this functional form, we note that the three regions of the curves in Figure 1 (the low-error plateau, the power-law region, and the high-error plateau) can be described by three power laws: two plateaus with exponent zero and one intermediate region with exponent γ. A functional family that arises frequently in the context of systems that exhibit different power-law regions is the *rational family*. The particular family member we consider is as follows:[3]

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

This function's shape is controlled by ϵ_{np}, $\epsilon^\uparrow$, γ, and p (visualized in Figure 1, right). ϵ_{np} and $\epsilon^\uparrow$ are the values of the low and high-error plateaus. γ is the slope of the power-law region on logarithmic axes. p controls the density at which the high-error plateau transitions to the power-law region.

Fitting. To fit $\hat{\epsilon}(\epsilon_{np}, d \mid l, w, n)$ to the actual data $\epsilon(d, l, w, n)$, we estimate values for the free parameters $\epsilon^\uparrow$, γ, and p by minimizing the relative error $\delta \triangleq \frac{\hat{\epsilon}(\epsilon_{np}, d|l,w,n) - \epsilon(d,l,w,n)}{\epsilon(d,l,w,n)}$ using least

[3]The expression $\left\| \frac{d-ja}{d-jb} \right\|^{\gamma} = \left(\frac{d^2+a^2}{d^2+b^2}\right)^{\frac{\gamma}{2}}$ meaning Eq. 1 can be rewritten as $\epsilon_{np}\left(\frac{d^2+p^2\left(\epsilon^\uparrow/\epsilon_{np}\right)^{2/\gamma}}{d^2+p^2}\right)^{\gamma/2}$

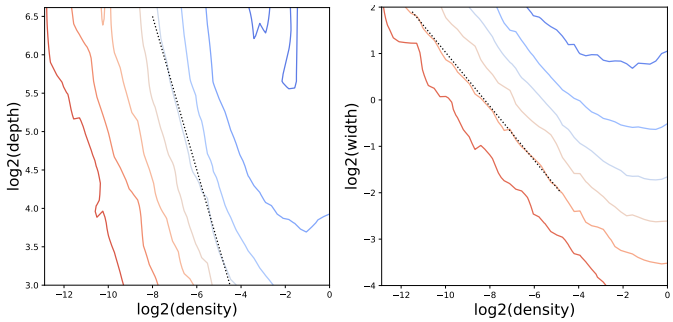

Figure 3: Projections of $\epsilon(d, l, w, n)$ onto two-dimensional planes for the CIFAR-10 ResNets, showing contours of constant error. Aside from the highest densities, the contours have linear slopes on the logarithmic axes. (left) The density/depth plane. (right) The density/width plane.

squares regression. The fit is performed separately for each configuration (l, w, n) for all 30–40 densities, resulting in per-configuration estimates of $\hat{\epsilon}^{\uparrow}$, $\hat{\gamma}$, and $\hat{p}$.

Evaluating fit. In Figure 2 (left), we plot the actual error[4] $\epsilon(d, l, w, n)$ and the estimated error $\hat{\epsilon}(\epsilon_{np}, d \mid l, w, n)$ for CIFAR-10 ResNets of varying widths. Qualitatively, our estimated error appears to closely follow the actual error. The most noticeable deviations occur at large densities, where the error dips slightly when pruning whereas we treat it as flat (see Section 5).

Quantitatively, we measure the extent to which estimated error departs from the actual error using the mean μ and standard deviation σ of the relative deviation δ. Figure 2 (center) compares the estimated and actual errors for the networks in Figure 2 (left); Figure 2 (right) shows the same comparison for all configurations of l, w, and n on CIFAR-10 and the resulting more than 4000 pruned models. The relative deviation on all configurations has mean $\mu < 2\%$ and standard deviation $\sigma < 4\%$; this means that, if the actual error is 10%, the estimated error is $9.8 \pm 0.4\%$ ($\hat{\epsilon} = (1 - \delta)\epsilon$).

4 Jointly Modeling Error Across All Dimensions

In Section 3, we found a functional form $\hat{\epsilon}(\epsilon_{np}, d \mid l, w, n)$ (Equation 1) that accurately predicts the error when pruning a *specific* member of a network family (with depth l and width w) trained with a dataset of size n. The parameters governing Equation 1 ($\epsilon_{\uparrow}$, p, and γ) were allowed to vary between different configurations and depend on l, w, n. However, we are interested in a single *joint* scaling law $\hat{\epsilon}(\epsilon_{np}, d, l, w, n)$ that, given the unpruned network error $\epsilon_{np}(l, w, n)$, accurately predicts error across *all* dimensions we consider: all members of a network family that vary in depth and width, all densities, and all dataset sizes. Importantly, the parameters of such a *joint* scaling law must be constants as a function of all dimensions. In this section, we develop this joint scaling law.

The error-preserving invariant. Our desired scaling law $\hat{\epsilon}(\epsilon_{np}, d, l, w, n)$ will be a four-dimensional function of d, w, l, and n. To develop such a functional form, we study the interdependence between density and depth or width by examining two-dimensional projections of the actual error $\epsilon(d, l, w, n)$ (Figure 3). These plots display contours of constant error as density and depth or width vary.

Consider the projection onto the plane of density and depth (Figure 3, left). The constant-error contours are linear except for in the densest networks, meaning each contour traces a power-law relationship between d and l. In other words, we can describe all combinations of densities and widths that produce error ϵ_v using $l^{\phi} d = v$, where v is a constant at which network error is ϵ_v and ϕ is the slope of the contour on the logarithmic axes. The contours of density and width also have this pattern (Figure 3, right), meaning we can describe a similar relationship $w^{\psi} d = v'$. Finally, we can combine these observations about depth and width into the expression $l^{\phi} w^{\psi} d = v''$.

We refer to the expression $l^{\phi} w^{\psi} d$ as the *error-preserving invariant*, and we denote it m^*. This invariant captures the observation that there exist many interchangeable combinations of depth, width, and density that achieve the same error and tells us which combinations do so. We note, for example, that networks of vastly different densities are freely interchangeable with no effect on error.

[4] We compute the error as the mean across three replicates with different random seeds and dataset subsamples.

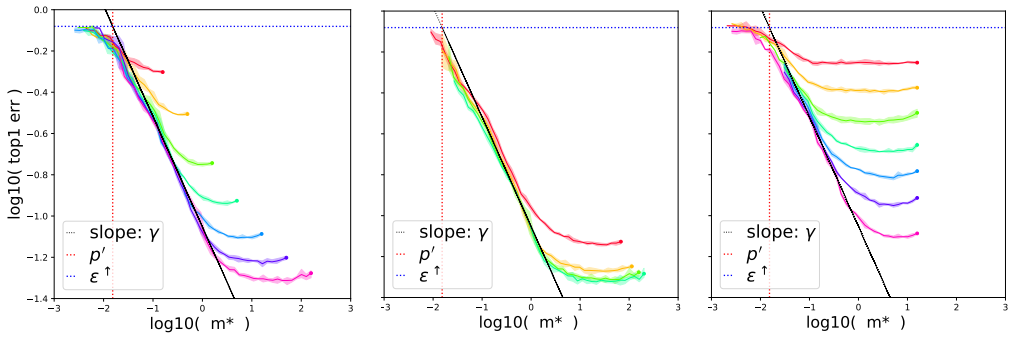

Figure 4: Relationship between m^* and error when pruning CIFAR-10 ResNets and varying w (left, $l = 20$, $n = N$), l (center, $w = 1$, $n = N$), n (right, $l = 20$, $w = 1$). γ, $\epsilon^{\uparrow}$, and p' are annotated.

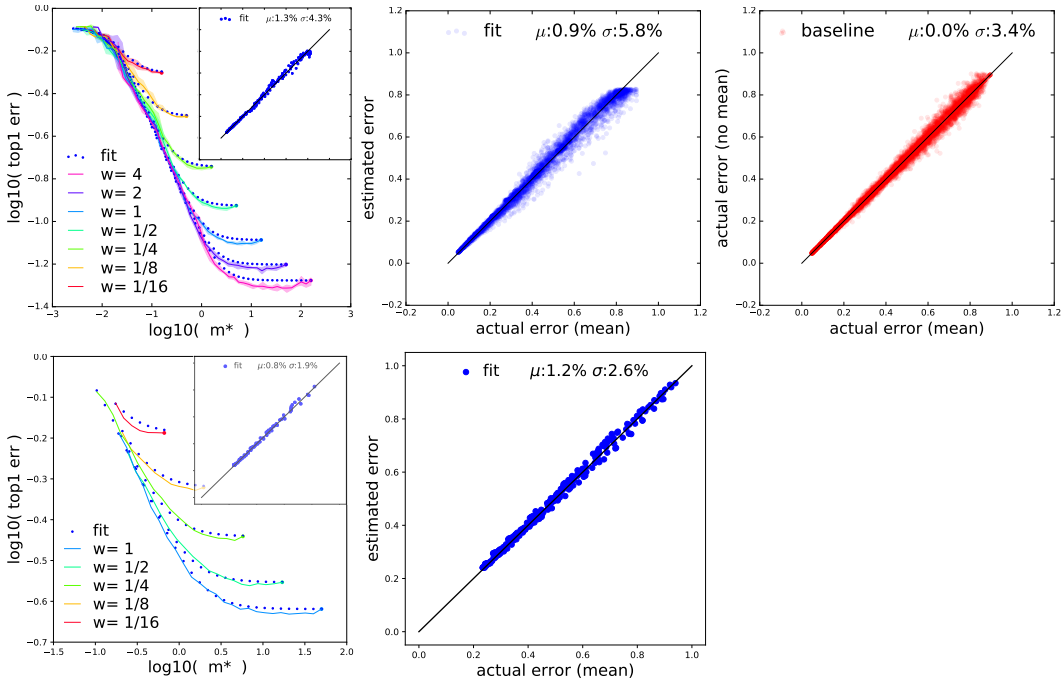

Figure 5: (left column) Actual error (solid) and estimated error using the joint scaling law (blue dots) versus $m*$ when varying width. (center) Estimated versus mean actual error for all configurations (d, w, l, n) for CIFAR-10, (d, w, n) for ImageNet. (right column) The variation in error when running the same experiment on CIFAR-10 three times with different random seeds.

Functional form. The invariant allows us to convert the functional form $\hat{\epsilon}(\epsilon_{np}, d \mid l, w, n)$ for a specific l, w, and n from Section 3 into a joint functional form $\hat{\epsilon}(\epsilon_{np}, d, l, w, n)$ for all l, w, and n. Rewriting the definition of the invariant, $d = \frac{m^*}{l^\phi w^\psi}$. We can substitute this for d in the functional form from Section 3. Finally, by rewriting p as $\frac{p'}{l^\phi w^\psi}$ and canceling, we arrive at the expression:

$$\hat{\epsilon}(\epsilon_{np}, d \mid l, w, n) = \epsilon_{np} \left\| \frac{m^* - jp' \left(\frac{\epsilon^{\uparrow}}{\epsilon_{np}}\right)^{\frac{1}{\gamma}}}{m^* - jp'} \right\|^{\gamma} = \epsilon_{np} \left\| \frac{l^\phi w^\psi d - jp' \left(\frac{\epsilon^{\uparrow}}{\epsilon_{np}}\right)^{\frac{1}{\gamma}}}{l^\phi w^\psi d - jp'} \right\|^{\gamma} = \hat{\epsilon}(\epsilon_{np}, d, l, w, n) \quad (2)$$

which is the joint functional form $\hat{\epsilon}(\epsilon_{np}, d, l, w, n)$ of all four dimensions d, l, w, and n. Critically, the free parameters $e^{\uparrow}$, p', and γ are constants shared across all possible values of d, l, w, and n. We see this by examining the relationship between m^* and generalization error of pruned networks as we vary depth, width, and dataset size (Figure 4). Across all configurations, $e^{\uparrow}$ (the error of the high-error plateau), γ (the slope of the power-law region) and p' (the value of m^* where the high-error plateau transitions to the power-law region) are the same.

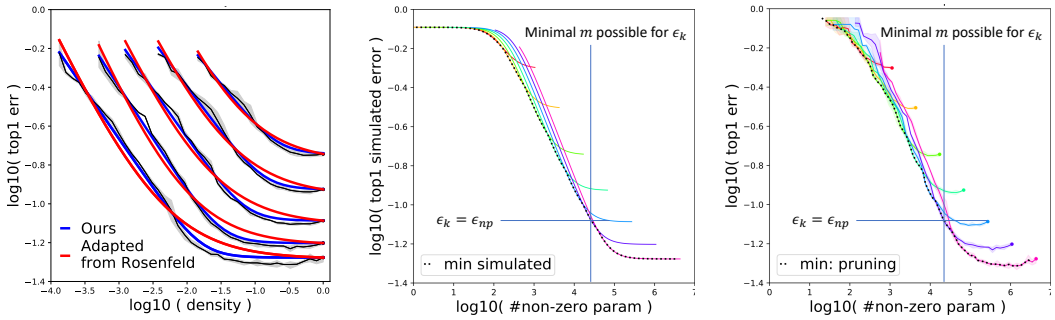

Figure 6: (left) Error versus density for the CIFAR-10 ResNets as width varies. (grey) Actual error, (blue) error of our scaling law and (red) the scaling law adapted from Rosenfeld et al. [2020]. (center) Estimated error from our scaling law as width varies for the CIFAR-10 ResNets. In dotted black is the minimal number of parameters for each error ϵ_k. (right) Same as center but using actual error.

Fitting. To fit $\hat{\epsilon}(\epsilon_{np}, d, l, w, n)$ to the actual data $\epsilon(d, l, w, n)$, we estimate values for the free parameters $\epsilon^{\uparrow}$, γ, p', ϕ and ψ by minimizing the relative error $\delta \triangleq \frac{\hat{\epsilon}(\epsilon_{np},d,l,w,n)-\epsilon(d,l,w,n)}{\epsilon(d,l,w,n)}$ using least squares regression. The fit is performed jointly over all configurations of d, l, w, and n (more than 4,000 points in all), resulting in joint estimates of $\hat{\epsilon}^{\uparrow}$, $\hat{\gamma}$, $\hat{p}$, $\hat{\phi}$ and $\hat{\psi}$. One can also perform a partial fit for a subset of the dimensions (e.g., just d, l, and n) by omitting ϕ and/or ψ.

Evaluating fit. In Figure 5 (left column), we plot the actual error $\epsilon(d, l, w, n)$ and the estimated error $\hat{\epsilon}(\epsilon_{np}, d, l, w, n)$ when varying width for the CIFAR-10 ResNets and ImageNet ResNets. As in Section 3, our estimated error appears to closely follow the actual error, with noticeable deviations mainly at high densities where error dips below ϵ_{np}. For additional experiments, including partial scaling on subsets of the dimensions, see Appendix E.

We again quantify the fit of the estimated error using the mean μ and standard deviation σ of the relative deviation δ. Figure 5 (center column) compares the estimated and actual errors for the joint scaling laws, and the insets in Figure 5 (left column) do the same when fitting for just d and w. The relative deviation on the joint scaling laws for the CIFAR-10 and Imagenet networks has a mean $\mu < 2\%$ and standard deviation of $\sigma < 6\%$.

To contextualize these results, Figure 5 (right column) quantifies the variation in error we see over multiple trials of the CIFAR-10 experiments due to using different random seeds. It plots the minimum, maximum, and mean errors across the three trials we ran.[5] The variation across trials has a standard deviation of $\sigma = 3.4\%$, sizeable relative to the estimation error of $\sigma = 5.8\%$ for the joint scaling law. This indicates that a significant portion of our error may stem from measurement noise.

5 Principles for Selecting a Functional Family

We have shown that our proposed functional form $\hat{\epsilon}(d, l, w, n)$ accurately approximates the generalization error when pruning a family of neural networks. In this section, we discuss some of the key criteria that led us to select this particular functional form and opportunities for further refinement.

Criterion 1: Transitions. In Section 3, we observe that, when pruning a neural network, error has a low-error plateau, a power-law region, and a high-error plateau. Between these regions are *transitions* where error varies smoothly from one region to the next. Matching the shape of these transitions was a key consideration for selecting our function family. To illustrate the importance of properly fitting the transitions, Figure 6 (left) shows two possible functional families for fitting the relationship between density and error for CIFAR-10 ResNets. Actual error is in gray, and the functional form from Section 3 is in blue. In red is the fit for a functional form adapted from the one that Rosenfeld et al. [2020] use to model the relationship between width and error. The difference between these functional families is the way they model transitions, and the one we choose in this paper better models the transitions in our setting. For further discussion of this comparison, see Appendix D.

Criterion 2: A small number of interpretable parameters. Selecting a functional form is not merely a curve-fitting exercise. We aim to find the underlying structure that governs the relationships

[5] We only ran a single trial of the ImageNet experiments due to the significant cost of collecting data.

between d, l, w, n, and error. As such, our functional form should have a small number of parameters that are *interpretable*. In the functional form that we find (Equation 2), each parameter has a clear meaning. The parameters $\epsilon^{\uparrow}$, p', and γ control the high-error plateau, the transition to the power-law region, and the slope of the power-law region, respectively. Complementary, ϕ and ψ control the interchangeability of width and depth with density. We approximate error over multiple orders of magnitude and over 4,000 distinct configurations with just five parameters, indicating that we have managed to distill key information about the behavior of these networks into our functional form.

Sources of systemic error. By seeking to minimize the number of parameters in our functional form, we leave some phenomena unmodeled. In particular, there are two phenomena we have chosen *not* to model that introduce systemic error. First, the low-error plateau is not a plateau. Error often improves slightly at high densities before returning to ϵ_{np} during the transition to the power-law region. Our model treats the region as flat and treats error as monotonically increasing as density decreases. This source of error accounts for a bias of $\sim 1\%$ relative error in our estimation (Appendix F). Second, we model both transitions (between the power-law region and each plateau) with a single shape and the same transition rate. If we treated each transition separately and used higher-order terms in the rational form, we could potentially reduce some of the residual error in our estimation.

6 Implications and Conclusions

Our main contribution is a functional form $\hat{\epsilon}(\epsilon_{np}, d, l, w, n)$ that accurately predicts the error when pruning members of a network family. There are several broader implications of our ability to characterize pruning in this way. The mere existence of this functional form means there is indeed structure to the way pruning affects error. Although prior work [Cai et al., 2019] has implicitly relied on such structure, we are the first to propose an explicit functional form describing it. This functional form enables a framework in which we can reason conceptually and analytically about pruning. In doing so, we can make new observations about pruning that are non-obvious or costly to exhaustively demonstrate empirically. For example, recall our motivating question:

Given a family of neural networks, which should we prune (and by how much) to obtain the network with the smallest parameter-count such that its error does not exceed some threshold ϵ_k?

This is an optimization problem: find the configuration of d, l, and w that minimizes parameter-count subject to an error constraint. The parameter-count is proportional to dlw^2, so we can solve this optimization problem analytically without running any further experiments.

Using this approach, we can derive a useful insight. In the pruning literature, it is standard practice to report the minimum density at which the pruned network can match the error $\epsilon_{np}(l, w)$ of the unpruned network [Han et al., 2015]. However, our scaling law suggests that this is not the smallest model that achieves error $\epsilon_{np}(l, w)$. Instead, it is better to train a larger network with depth l' and width w' and prune until error reaches $\epsilon_{np}(l, w)$, even if that results in error well above $\epsilon_{np}(l', w')$ (similar to the empirical finding of Li et al. [2020] on NLP tasks).

Figure 6 (center) illustrates this behavior: it shows the error predicted by our scaling law for CIFAR-10 ResNets with varying widths. The dotted black line shows the minimal parameter-count at which we predict it is possible to achieve each error. Importantly, none of the low-error plateaus intersect this black dotted line, meaning a model cannot be minimal until it has been pruned to the point where it increases in error. This occurs because the transitions of our functional form are gradual. On the other hand, if we start with a model that is too large, it will no longer be on the black line when it has been pruned to the point where its error reaches $\epsilon_{np}(l, w)$. This occurs because error decreases as a function of m^* rather than parameter-count. In Figure 6 (right), we plot the same information from the actual CIFAR-10 data. We see that the same phenomena occur in practice. Quantitatively, the estimated versus actual parameter count at optimum differs by up to 25%.

Looking ahead, there are several opportunities to extend the generality—and thereby, utility—of this framework. For example, by studying other classes of networks and tasks, we can confirm that our scaling law applies in these settings or revise it to include this new data. By studying other pruning methods, we can further understand which aspects of our framework (e.g., the scaling law itself, the invariant, the design approach) generalize. Finally, we can explore fitting our scaling law to small-scale settings and extrapolating upwards to larger networks and datasets, which would make it possible to reason analytically without ever having to train a large-scale network.

Broader Impact

We do not believe that our work will create new societal harm. Our work is scientific in nature: it aims to uncover and describe empirical phenomena that emerge in deep learning. Although better understanding of neural networks can potentially make it easier to use this technology for societal harm, we do not believe that our paper poses any unusual risks. The biggest societal harm of this work (and any empirical deep learning work) is the energy required to run these experiments.

However, we also believe this work has the potential for societal benefits. As the computational demands of deep learning continue to increase, we risk (1) impact on the environment, (2) concentration of power by those who have resources, and (3) slower innovation. Our work is part of a larger effort to uncover structure in the way that large-scale neural networks behave as we change their structure (or the structure of the optimization problem) [Hestness et al., 2017, Tan and Le, 2019, Rosenfeld et al., 2020, Kaplan et al., 2020]. This collective effort seeks to replace the extraordinarily expensive [Strubell et al., 2019] process of network architecture search and hyperparameter search with a more principled approach that requires training far fewer networks. In doing so, we can mitigate the emerging societal risks caused by deep learning's enormous computational demands.

Acknowledgments and Disclosure of Funding

Jonathan Rosenfeld was funded by the Efficient NN Model Scaling grant from Analog Devices Incorporated. This work was partially supported by NSF grants CCF-1563880 and IIS-1607189.

We gratefully acknowledge the support of IBM, which provided us with access to the GPU resources necessary to conduct experiments on CIFAR-10 through the MIT-IBM Watson AI Lab. In particular, we express our gratitude to David Cox and John Cohn.

We gratefully acknowledge the support of Google, which provided us with access to the TPU resources necessary to conduct experiments on ImageNet through the TensorFlow Research Cloud. In particular, we express our gratitude to Zak Stone.

This work was supported in part by cloud credits from the MIT Quest for Intelligence.

We thank Yonatan Belinkov and Roy Shilkrot for their constructive comments.

References

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Conference on Machine Learning and Systems*, 2020.

Han Cai, Chuang Gan, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, 2020.

Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Steven A. Janowsky. Pruning versus clipping in neural networks. *Phys. Rev. A*, 39:6600–6603, Jun 1989. doi: 10.1103/PhysRevA.39.6600. URL https://link.aps.org/doi/10.1103/PhysRevA.39.6600.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E Gonzalez. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *arXiv preprint arXiv:2002.11794*, 2020.

Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

Alex Renda, Jonathan Frankle, and Michael Carbin. Comparing rewinding and fine-tuning in neural network pruning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1gSj0NKvB.

Jonathan S. Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryenvpEKDr.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

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

A Formal Statement of Pruning Algorithm

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

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 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.

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.

We train CIFAR-10 ResNets 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. Data is augmented by normalizing, randomly flipping left and right, and randomly shifting by up to four pixels in any direction (and cropping afterwards). All CIFAR-10 networks are trained on GPUs.

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. All ImageNet networks are trained on GPUs.

Our codebase was written in PyTorch.

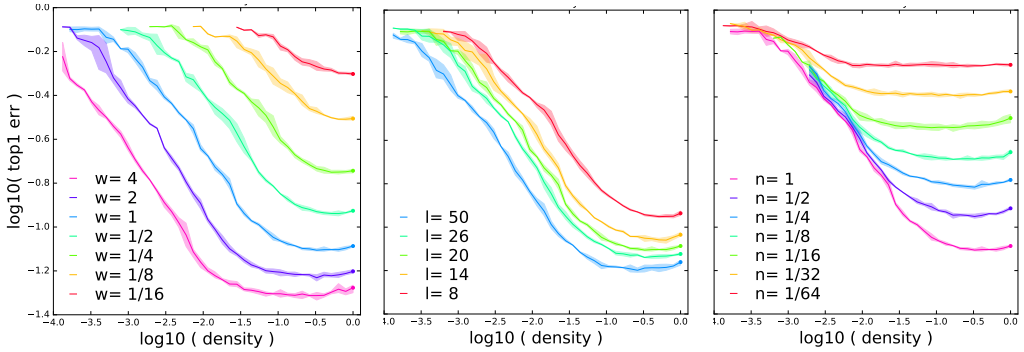

Figure 7: Relationship between density and error when pruning CIFAR-10 ResNets and varying w (left, $l = 20$, $n = N$), l (center, $w = 1$, $n = N$), n (right, $l = 20$, $w = 1$)

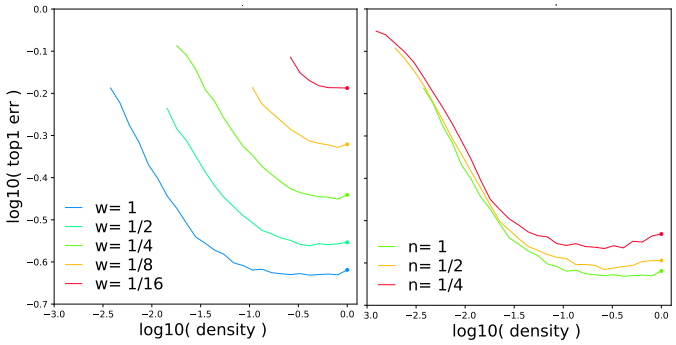

Figure 8: Relationship between density and error when pruning Imagenet ResNet-50 and varying w (left, $n = N$), and n (right, $w = 1$)

C Full Data for Key Observations

In this appendix, we show that our observations from Section 3 hold when varying all dimensions (depth, width, and dataset size) on both CIFAR-10 and ImageNet. Figure 7 shows the error versus density when changing width (left) depth (center) and data (right). In Figure 8, we similarly show the dependency of the error on density for Imagenet when varying width (left) and dataset size (right).

In Figure 7, we observe that all curves have a similar slope in the power-law region. In the notation to follow in Equation 6, 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 well approximated by a constant. In contrast, the transition from high-error plateau to the power-law region is clearly not constant as a function of density. Section 4 finds exactly this dependency of the transition parameter p.

D Comparison of Pruning Versus Non-pruning Scaling Laws

In this appendix, we contrast the behavior of the error when pruning with the bahvior of the error in the non-pruning setting. Hestness et al. [2017] show the the error follows a saturating powerlaw 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:

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

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

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 9 (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 9 (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.

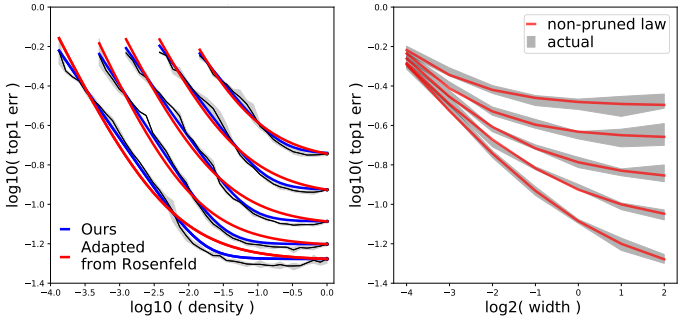

Figure 9: (Left) Error versus density for different widths. In blue is the fit eq. 2 follows closely the actual error (black) while in red is the fit for the adapted from Rosenfeld et al. [2020] which deviates noticeably in comparison. (Right) error of non-pruned networks versus width for different data, 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.

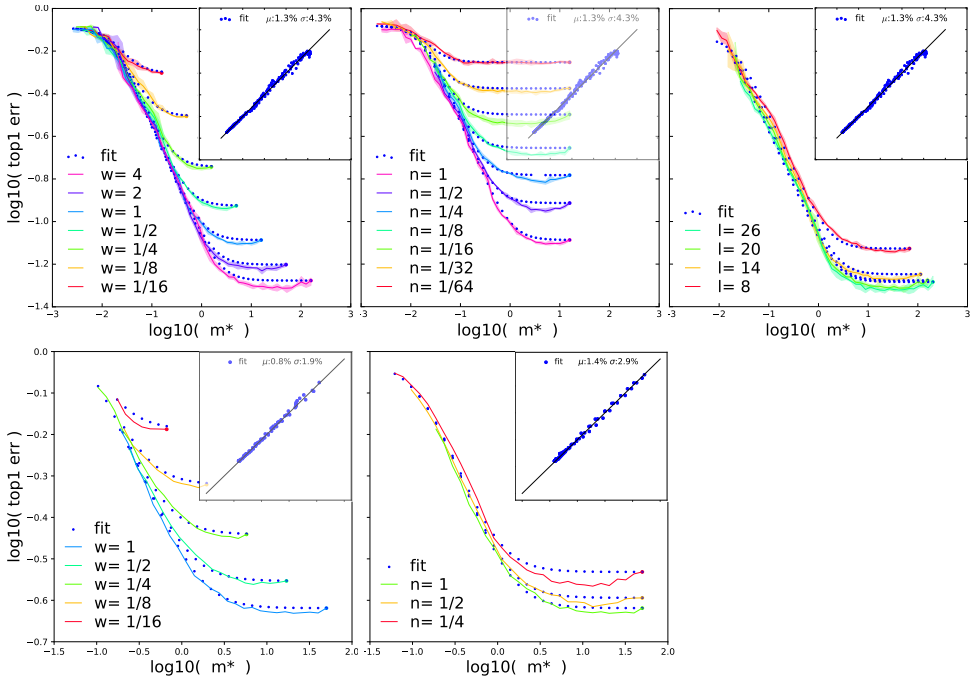

Figure 10: Top- CIFAR-10, Bottom- Imagenet. Actual error (solid) and estimated error using the joint scaling law (blue dots) versus m^*. (left) when varying width, explicitly, ϕ is not estimated. (right) when varying depth, ψ is not estimated. (center) when varying only data - width and depth are not varies, and ψ and ϕ are not estimated.

E Additional fit results

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. 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 10 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.

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. 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.

F 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 $\sim 1\%$.