Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data

Charles H. Martin[*] Tongsu (Serena) Peng[†] Michael W. Mahoney[‡§]

Abstract

In many applications, one works with neural network models trained by someone else. For such pretrained models, one may not have access to training data or test data. Moreover, one may not know details about the model, e.g., the specifics of the training data, the loss function, the hyperparameter values, etc. Given one or many pretrained models, it is a challenge to say anything about the expected performance or quality of the models. Here, we address this challenge by providing a detailed meta-analysis of hundreds of publicly-available pretrained models. We examine norm based capacity control metrics as well as power law based metrics from the recently-developed Theory of Heavy-Tailed Self Regularization. We find that norm based metrics correlate well with reported test accuracies for well-trained models, but that they often cannot distinguish well-trained versus poorly-trained models. We also find that power law based metrics can do much better—quantitatively better at discriminating among series of well-trained models with a given architecture; and qualitatively better at discriminating well-trained versus poorly-trained models. These methods can be used to identify when a pretrained neural network has problems that cannot be detected simply by examining training/test accuracies.

1 Introduction

A common problem in machine learning (ML) is to evaluate the quality of a given model. A popular way to accomplish this is to train a model and then evaluate its training/testing error. There are many problems with this approach. The training/testing curves give very limited insight into the overall properties of the model; they do not take into account the (often large human and CPU/GPU) time for hyperparameter fiddling; they typically do not correlate with other properties of interest such as robustness or fairness or interpretability; and so on. A related problem, in particular in industrial-scale artificial intelligence (AI), arises when the model user is not the model developer. Then, one may not have access to the training data or the testing data. Instead, one may simply be given a model that has already been trained—a pretrained model—and need to use it as-is, or to fine-tune and/or compress it and then use it.

Naïvely—but in our experience commonly, among ML practitioners and ML theorists—if one does not have access to training or testing data, then one can say absolutely nothing about the quality of a ML model. This may be true in worst-case theory, but models are used in practice, and there is a need for a practical theory to guide that practice. Moreover, if ML is to become an industrial process, then that process will become compartmentalized in order to scale: some

[*]Calculation Consulting, 8 Locksley Ave, 6B, San Francisco, CA 94122, `charles@CalculationConsulting.com`.

[†]Calculation Consulting, 8 Locksley Ave, 6B, San Francisco, CA 94122, `serenapeng7@gmail.com`.

[‡]ICSI and Department of Statistics, University of California at Berkeley, Berkeley, CA 94720, `mmahoney@stat.berkeley.edu`.

[§]Corresponding author.

groups will gather data, other groups will develop models, and other groups will use those models. Users of models cannot be expected to know the precise details of how models were built, the specifics of data that were used to train the model, what was the loss function or hyperparameter values, how precisely the model was regularized, etc.

Moreover, for many large scale, practical applications, there is no obvious way to define an ideal test metric. For example, models that generate fake text or conversational chatbots may use a proxy, like perplexity, as a test metric. In the end, however, they really require human evaluation. Alternatively, models that cluster user profiles, which are widely used in areas such as marketing and advertising, are unsupervised and have no obvious labels for comparison and/or evaluation. In these and other areas, ML objectives can be poor proxies for downstream goals.

Most importantly, in industry, one faces unique practical problems such as determining whether one has enough data for a given model. Indeed, high quality, labeled data can be very expensive to acquire, and this cost can make or break a project. Methods that are developed and evaluated on any well-defined publicly-available corpus of data, no matter how large or diverse or interesting, are clearly not going to be well-suited to address problems such as this. It is of great practical interest to have metrics to evaluate the quality of a trained model—in the absence of training/testing data and without any detailed knowledge of the training/testing process. There is a need for a practical theory for pretrained models which can predict how, when, and why such models can be expected to perform well or poorly.

In the absence of training and testing data, obvious quantities to examine are the weight matrices of pretrained models, e.g., properties such as norms of weight matrices and/or parameters of Power Law (PL) fits of the eigenvalues of weight matrices. Norm-based metrics have been used in traditional statistical learning theory to bound capacity and construct regularizers; and PL fits are based on statistical mechanics approaches to deep neural networks (DNNs). While we use traditional norm-based and PL-based metrics, our goals are not the traditional goals. Unlike more common ML approaches, we do not seek a bound on the generalization (e.g., by evaluating training/test errors), we do not seek a new regularizer, and we do not aim to evaluate a single model (e.g., as with hyperparameter optimization). Instead, we want to examine different models across common architecture series, and we want to compare models between different architectures themselves. In both cases, one can ask whether it is possible to predict trends in the quality of pretrained DNN models without access to training or testing data.

To answer this question, we provide a detailed empirical analysis, evaluating quality metrics for pretrained DNN models, and we do so at scale. Our approach may be viewed as a statistical meta-analysis of previously published work, where we consider a large suite of hundreds of publicly-available models, mostly from computer vision (CV) and natural language processing (NLP). By now, there are many such state-of-the-art models that are publicly-available, e.g., hundreds of pretrained models in CV (≥ 500) and NLP (≈ 100).[1] For all these models, we have no access to training data or testing data, and we have no specific knowledge of the training/testing protocols. Here is a summary of our main results. First, norm-based metrics do a reasonably good job at predicting quality trends in well-trained CV/NLP models. Second, norm-based metrics may give spurious results when applied to poorly-trained models (e.g., models trained without enough data, etc.). For example, they may exhibit what we call Scale Collapse for these models. Third, PL-based metrics can do much better at predicting quality trends in pretrained CV/NLP models. In particular, a weighted PL exponent (weighted by the log of the spectral norm of the corresponding layer) is quantitatively better at discriminating among a series of well-trained versus very-well-trained models within a given architecture series; and the (unweighted)

[1]When we began this work in 2018, there were fewer than tens of such models; now in 2020, there are hundreds of such models; and we expect that in a year or two there will be an order of magnitude or more of such models.

average PL exponent is qualitatively better at discriminating well-trained versus poorly-trained models. Fourth, PL-based metrics can also be used to characterize fine-scale model properties, including what we call layer-wise Correlation Flow, in well-trained and poorly-trained models; and they can be used to evaluate model enhancements (e.g., distillation, fine-tuning, etc.). Our work provides a theoretically-principled empirical evaluation—by far the largest, most detailed, and most comprehensive to date—and the theory we apply was developed previously [1, 2, 3]. Performing such a meta-analysis of previously-published work is common in certain areas, but it is quite rare in ML, where the emphasis is on developing better training protocols.

2 Results

After describing our overall approach, we study in detail three well-known CV architecture series (the VGG, ResNet, and DenseNet series of models). Then, we look in detail at several variations of a popular NLP architecture series (the OpenAI GPT and GPT2 series of models), and we present results from a broader analysis of hundreds of pretrained DNN models.

2.1 Overall approach

Consider the objective/optimization function (parameterized by $\mathbf{W}_l$s and $\mathbf{b}_l$s) for a DNN with L layers, and weight matrices $\mathbf{W}_l$ and bias vectors $\mathbf{b}_l$, as the minimization of a general loss function $\mathcal{L}$ over the training data instances and labels, $\{\mathbf{x}_i, y_i\} \in \mathcal{D}$. For a typical supervised classification problem, the goal of training is to construct (or learn) $\mathbf{W}_l$ and $\mathbf{b}_l$ that capture correlations in the data, in the sense of solving

$$\underset{\mathbf{W}_l,\mathbf{b}_L}{\text{argmin}} \sum_{i=1}^{N} \left(\mathcal{L}(E_{DNN}(\mathbf{x}_i) - y_i)\right), \tag{1}$$

[michael: how precisely to change eqn for R3 comment] where the loss function $\mathcal{L}(\cdot)$ can take on a myriad of forms [4], and where the energy (or optimization) landscape function

$$E_{DNN} = f(\mathbf{x}_i; \mathbf{W}_1, \ldots, \mathbf{W}_L, \mathbf{b}_1, \ldots, \mathbf{b}_L) \tag{2}$$

[michael: sentences around here probably need to be changed in light of change to that eqn R3 wanted] depends parametrically on the weights and biases. [michael: FROM V1: See comments for math equations.] For a trained model, E_{DNN} does not explicitly depend on the data, but rather it maps data instance vectors ($\mathbf{x}_i$ values) to predictions (y_i labels). [michael: FROM V1: Mark found that sentence confusing.] Therefore, one can analyze E_{DNN} in the absence of any training or test data.

Test accuracies have been reported online for publicly-available pretrained pyTorch models [5]. These models have been trained and evaluated on labeled data $\{\mathbf{x}_i, y_i\} \in \mathcal{D}$, using standard techniques. We do not have access to this data, and we have not trained any of the models ourselves. Our methodological approach is thus similar to a statistical meta-analysis, common in biomedical research, but uncommon in ML. Computations were performed with the publicly-available `WeightWatcher` tool (version 0.2.7) [6]. To be fully reproducible, we only examine publicly-available, pretrained models, and we provide all Jupyter and Google Colab notebooks used in an accompanying github repository [7]. See the Supplementary Information for details.

Metrics for DNN Weight Matrices. Our approach involves analyzing individual DNN weight matrices, for (depending on the architecture) fully-connected and/or convolutional layers.

Each DNN layer contains one or more layer 2D $N_l \times M_l$ weight matrices, $\mathbf{W}_l$, or pre-activation maps, $\mathbf{W}_{i,l}$, e.g., extracted from 2D Convolutional layers, where $N > M$. See the Supplementary Information for details. (We may drop the i and/or i,l subscripts below.) The best performing quality metrics depend on the norms and/or spectral properties of each weight matrix, $\mathbf{W}$, and/or, equivalently, it's empirical correlation matrix, $\mathbf{X} = \mathbf{W}^T\mathbf{W}$. To evaluate the quality of state-of-the-art DNNs, we consider the following metrics:

$$\text{Frobenius Norm: } \|\mathbf{W}\|_F^2 = \|\mathbf{X}\|_F = \sum\nolimits_{i=1}^{M} \lambda_i \tag{3}$$

$$\text{Spectral Norm: } \|\mathbf{W}\|_\infty^2 = \|\mathbf{X}\|_\infty = \lambda_{max} \tag{4}$$

$$\text{Weighted Alpha: } \hat{\alpha} = \alpha \log \lambda_{max} \tag{5}$$

$$\alpha\text{-Norm (or } \alpha\text{-Shatten Norm): } \|\mathbf{W}\|_{2\alpha}^{2\alpha} = \|\mathbf{X}\|_\alpha^\alpha = \sum\nolimits_{i=1}^{M} \lambda_i^\alpha. \tag{6}$$

To perform diagnostics on potentially-problematic DNNs, we will decompose $\hat{\alpha}$ into its two components, α and λ_{max}. Here, λ_i is the i^{th} eigenvalue of the $\mathbf{X}$, λ_{max} is the maximum eigenvalue, and α is the fitted PL exponent. These eigenvalues are squares of the singular values σ_i of $\mathbf{W}$, $\lambda_i = \sigma_i^2$. All four metrics can be computed easily from DNN weight matrices. The first two metrics are well-known in ML. The last two metrics deserve special mention, as they depend on an empirical parameter α that is the PL exponent that arises in the recently-developed Heavy Tailed Self Regularization (HT-SR) Theory [1, 2, 3].

Overview of Heavy-Tailed Self-Regularization. In the HT-SR Theory, one analyzes the eigenvalue spectrum, i.e., the Empirical Spectral Density (ESD), of the associated correlation matrices [1, 2, 3]. From this, one characterizes the amount and form of correlation, and therefore implicit self-regularizartion, present in the DNN's weight matrices. For each layer weight matrix $\mathbf{W}$, of size $N \times M$, construct the associated $M \times M$ (uncentered) correlation matrix $\mathbf{X}$. Dropping the L and l, i indices, one has

$$\mathbf{X} = \frac{1}{N}\mathbf{W}^T\mathbf{W}.$$

If we compute the eigenvalue spectrum of $\mathbf{X}$, i.e., λ_i such that $\mathbf{X}\mathbf{v}_i = \lambda_i \mathbf{v}_i$, then the ESD of eigenvalues, $\rho(\lambda)$, is just a histogram of the eigenvalues, formally written as $\rho(\lambda) = \sum_{i=1}^{M} \delta(\lambda - \lambda_i)$. Using HT-SR Theory, one characterizes the correlations in a weight matrix by examining its ESD, $\rho(\lambda)$. It can be well-fit to a truncated PL distribution, given as

$$\rho(\lambda) \sim \lambda^{-\alpha}, \tag{7}$$

which is (at least) valid within a bounded range of eigenvalues $\lambda \in [\lambda^{min}, \lambda^{max}]$.

The original work on HT-SR Theory considered a small number of NNs, including AlexNet and InceptionV3. It showed that for nearly every $\mathbf{W}$, the (bulk and tail) of the ESDs can be fit to a truncated PL, and that PL exponents α nearly all lie within the range $\alpha \in (1.5, 5)$ [1, 2, 3]. As for the mechanism responsible for these properties, statistical physics offers several possibilities [8, 9], e.g., self organized criticality [10, 11] or multiplicative noise in the stochastic optimization algorithms used to train these models [12, 13]. Alternatively, related techniques have been used to analyze correlations and information propogation in actual spiking neurons [14, 15]. Our meta-analysis does not require knowledge of mechanisms; and it is not even clear that one mechanism is responsible for every case. Crucially, HT-SR Theory predicts that smaller values of α should correspond to models with better correlation over multiple size scales and thus to better models. The notion of "size scale" is well-defined in physical systems, to which this style of analysis is usually applied, but it is less well-defined in CV and NLP applications. Informally, it would correspond to pixel groups that are at a greater distance in some metric, or between sentence parts that are at a greater distance in text. Relatedly, previous work observed that smaller exponents α correspond to more implicit self-regularization and better generalization, and that we expect a linear correlation between $\hat{\alpha}$ and model quality [1, 2, 3].

DNN Empirical Quality Metrics. For norm-based metrics, we use the average of the log norm, to the appropriate power. Informally, this amounts to assuming that the layer weight matrices are statistically independent, in which case we can estimate the model complexity $\mathcal{C}$, or test accuracy, with a standard Product Norm (which resembles a data dependent VC complexity),

$$\mathcal{C} \sim \|\mathbf{W}_1\| \times \|\mathbf{W}_2\| \times \cdots \times \|\mathbf{W}_L\|, \tag{8}$$

where $\|\cdot\|$ is a matrix norm. The log complexity,

$$\log \mathcal{C} \sim \log \|\mathbf{W}_1\| + \log \|\mathbf{W}_2\| + \cdots + \log \|\mathbf{W}_L\| = \sum\nolimits_l \log \|\mathbf{W}_l\|, \tag{9}$$

takes the form of an average Log Norm. For the Frobenius Norm metric and Spectral Norm metric, we can use Eqn. (9) directly (since, when taking $\log \|\mathbf{W}_l\|_F^2$, the 2 comes down and out of the sum, and thus ignoring it only changes the metric by a constant factor).

The Weighted Alpha metric is an average of α_l over all layers $l \in \{1, \ldots, l\}$, weighted by the size, or scale, or each matrix,

$$\hat{\alpha} = \frac{1}{L} \sum_l \alpha_l \log \lambda_{max,l} \approx \langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle, \tag{10}$$

where L is the total number of layer weight matrices. The Weighted Alpha metric was introduced previously [3], where it was shown to correlate well with trends in reported test accuracies of pretrained DNNs, albeit on a much smaller and more limited set of models than we consider here.

Based on this, in this paper, we introduce and evaluate the α-Shatten Norm metric,

$$\sum\nolimits_l \log \|\mathbf{X}_l\|_{\alpha_l}^{\alpha_l} = \sum\nolimits_l \alpha_l \log \|\mathbf{X}_l\|_{\alpha_l}. \tag{11}$$

For the α-Shatten Norm metric, α_l varies from layer to layer, and so in Eqn. (11) it cannot be taken out of the sum. For small α, the Weighted Alpha metric approximates the Log α-Shatten norm, as can be shown with a statistical mechanics and random matrix theory derivation; and the Weighted Alpha and α-Shatten norm metrics often behave like an improved, weighted average Log Spectral Norm.

Finally, although it does less well for predicting trends in state-of-the-art model series, e.g., as depth changes, the average value of α, i.e.,

$$\bar{\alpha} = \frac{1}{L} \sum_l \alpha_l = \langle \alpha \rangle, \tag{12}$$

can be used to perform model diagnostics, to identify problems that cannot be detected by examining training/test accuracies, and to discriminate poorly-trained models from well-trained models.

One determines α for a given layer by fitting the ESD of that layer's weight matrix to a truncated PL, using the commonly accepted Maximum Likelihood method [16, 17]. This method works very well for exponents between $\alpha \in (2, 4)$; and it is adequate, although imprecise, for smaller and especially larger α [18]. Operationally, α is determined by using the `WeightWatcher` tool [6] to fit the histogram of eigenvalues, $\rho(\lambda)$, to a truncated PL,

$$\rho(\lambda) \sim \lambda^\alpha, \quad \lambda \in [\lambda_{min}, \lambda_{max}], \tag{13}$$

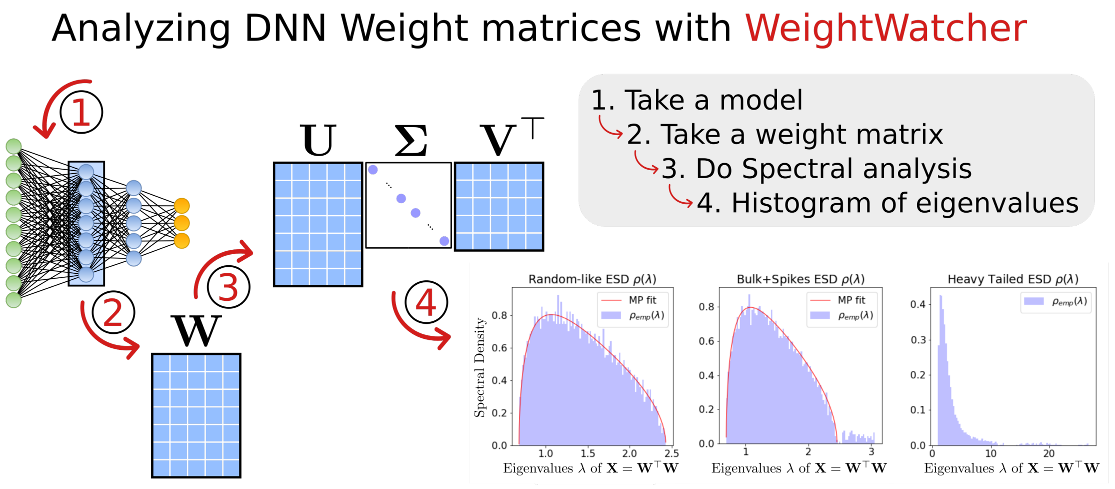


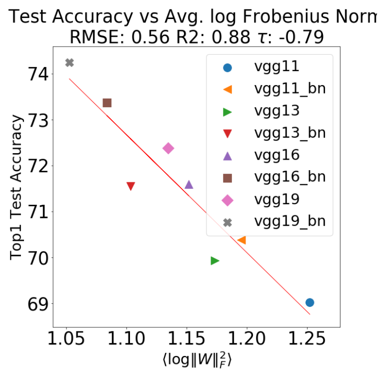
Figure 1: Schematic of analyzing DNN layer weight matrices $\mathbf{W}$. Given an individual layer weight matrix $\mathbf{W}$, from either a fully-connected layer or a convolutional layer, perform a Singular Value Decomposition (SVD) to obtain $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, and examine the histogram of eigenvalues of $\mathbf{W}^T\mathbf{W}$. Norm-based metrics and PL-based metrics (that depend on fitting the histogram of eigenvalues to a truncated PL) can be used to compare models. For example, one can analyze one layer of a pre-trained model, compare multiple layers of a pre-trained model, make comparisons across model architectures, monitor neural network properties during training, etc.

where λ_{max} is the largest eigenvalue of $\mathbf{X} = \mathbf{W}^T\mathbf{W}$, and where λ_{min} is selected automatically to yield the best (in the sense of minimizing the K-S distance) PL fit. Each of these quantities is defined for a given layer $\mathbf{W}$ matrix. See Figure 1 for an illustration.

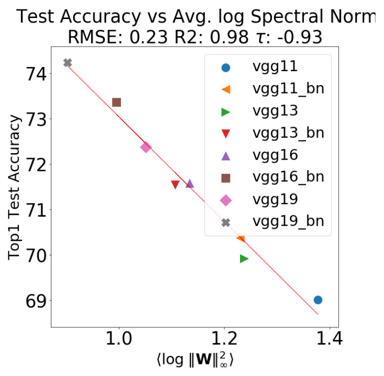
To avoid confusion, let us clarify the relationship between α and $\hat{\alpha}$. We fit the ESD of the correlation matrix $\mathbf{X}$ to a truncated PL, parameterized by 2 values: the PL exponent α, and the maximum eigenvalue λ_{max}. The PL exponent α measures the amount of correlation in a DNN layer weight matrix $\mathbf{W}$. It is valid for $\lambda \leq \lambda_{max}$, and it is scale-invariant, i.e., it does not depend on the normalization of $\mathbf{W}$ or $\mathbf{X}$. The λ_{max} is a measure of the size, or scale, of $\mathbf{W}$. Multiplying each α by the corresponding $\log \lambda_{max}$ weighs "bigger" layers more, and averaging this product leads to a balanced, Weighted Alpha metric $\hat{\alpha}$ for the entire DNN. We will see that for well-trained CV and NLP models, $\hat{\alpha}$ performs quite well and as expected, but for CV and NLP models that are potentially-problematic or less well-trained, metrics that depend on the scale of the problem can perform anomalously. In these cases, separating $\hat{\alpha}$ into its two components, α and λ_{max}, and examining the distributions of each, can be helpful.

2.2 Comparison of CV models

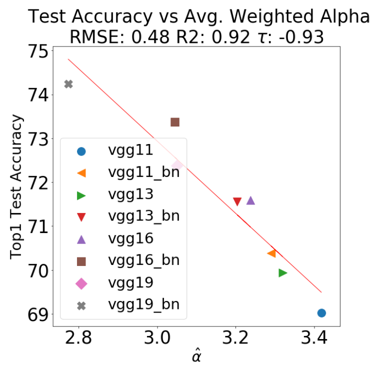
Each of the VGG, ResNet, and DenseNet series of models consists of several pretrained DNN models, with a given base architecture, trained on the full ImageNet [19] dataset, and each is distributed with the current open source pyTorch framework (version 1.4) [20]. In addition, we examine a larger set of ResNet models, which we call the ResNet-1K series, trained on the ImageNet-1K dataset [19] and provided on the OSMR Sandbox [5]. For these models, we first perform coarse model analysis, comparing and contrasting the four model series, and predicting trends in model quality. We then perform fine layer analysis, as a function of depth. This layer analysis goes beyond predicting trends in model quality, instead illustrating that PL-based metrics can provide novel insights among the VGG, ResNet/ResNet-1K, and DenseNet architectures.



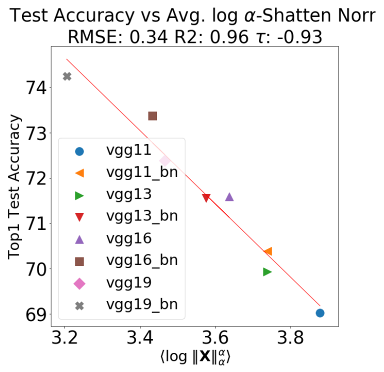
(a) Log Frobenius Norm, VGG



(b) Log Spectral Norm, VGG



(c) Weighted Alpha, VGG

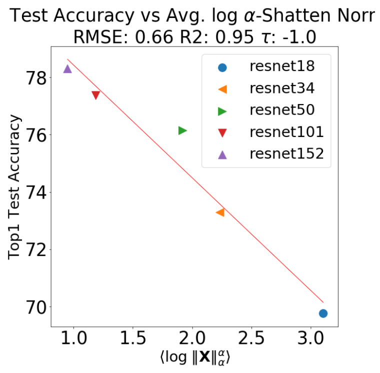


(d) Log α-Norm, VGG

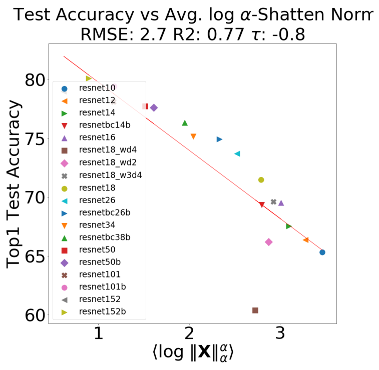
Figure 2: Comparison of Average Log Norm and Weighted Alpha quality metrics versus reported test accuracy for pretrained VGG models: VGG11, VGG13, VGG16, and VGG19, with and without Batch Normalization (BN), trained on ImageNet, available in pyTorch (v1.4). Metrics fit by linear regression, RMSE, R2, and the Kendal-tau rank correlation metric reported.

Average Quality Metrics versus Reported Test Accuracies. We examine the performance of the four quality metrics (Log Frobenius norm, Log Spectral norm, Weighted Alpha, and Log α-Norm) applied to each of the VGG, ResNet, ResNet-1K, and DenseNet series. Figure 2 plots the four quality metrics versus reported test accuracies [20],[2] as well as a basic linear regression line, for the VGG series. All four metrics correlate quite well with reported Top1 accuracies, with smaller norms and smaller values of $\hat{\alpha}$ implying better generalization (i.e., greater accuracy, lower error). [michael: The following, text and numbers, will need to be updated once we finalize the figures/tables.] The Log α-Norm metric ($\log \|\mathbf{W}\|_\alpha^\alpha$) performs best (RMSE of 0.42, R^2 of XXX, and Kendall-τ of XXX, see Table 1); and the Weighted Alpha metric ($\hat{\alpha} = \alpha \log \lambda_{max}$) performs second best (RMSE of 0.48, R^2 of XXX, and Kendall-τ of XXX, see Table 1). [michael: Put comment on why these three metrics, that HT-SR says that $\hat{\alpha}$ should exhibit good linear correlation, and a few comments on non-normality to address R1 concerns.] [charles: The Log Spectral Norm metric correlates very well with small, simpler data sets like the VGG and DenseNet architectures, however, when moving to the larger and more complex ResNet series, the PL-based metrics Log α-Norm ($\log \|\mathbf{W}\|_\alpha^\alpha$) and Weighted Alpha ($\hat{\alpha} = \alpha \log \lambda_{max}$) metrics perform better across all measures.]

[2]These test accuracies have been previously reported and made publicly-available by others. We take them as given. We do not attempt to reproduce/verify them, since we do not permit ourselves access to training/test data.



(a) ResNet, Log α-Norm



(b) ResNet-1K, Log α-Norm

Figure 3: Comparison of Average α-Norm quality metric versus reported Top1 test accuracy for the ResNet and ResNet-1K pretrained (pyTorch) models. [michael: FROM V1: Symbols and text are way to small on these and other figures; axes/ticks way too small; need different symbols, not just red versus green; etc.] [charles: See changes]

[michael: The following, text and numbers, will need to be updated once we finalize the figures/tables.] Overall, all metrics perform relatively well; the Weighted Alpha metric performs best; and the Log α-Norm metric performs second best. See Table 1 for a summary of results for Top1 accuracies for all four metrics for the VGG, ResNet, ResNet-1K, and DenseNet series. Similar results are obtained for the Top5 accuracies. These model series are all very well-trodden, and our results indicate that norm-based metrics and PL-based metrics can both distinguish among a series of well-trained versus very-well-trained models, with PL-based metrics performing somewhat (i.e., quantitatively) better. The DenseNet series has similar behavior. (These and many other such plots can be seen on our publicly-available repo.)

To examine how the four quality metrics depend on data set size, we look at results on ResNet versus ResNet-1K. Figure 3 compares the Log α-Norm metric for the full ResNet model, trained on the full ImageNet dataset, against the ResNet-1K model, trained on a much smaller ImageNet-1K data set. [michael: The following, text and numbers, will need to be updated once we finalize the figures/tables.] The Log α-Norm is much better than the Log Frobenius/Spectral norm metrics (although, as Table 1 shows, it is slightly worse than the Weighted Alpha metric). The ResNet series has strong correlation (RMSE of 0.66, R^2 of XXX, and Kendall-τ of XXX), whereas the ResNet-1K series also shows good but weaker correlation (much larger RMSE of 1.9, R^2 of XXX, and Kendall-τ of XXX). (Other metrics exhibit similar behavior.) The higher quality data set shows a better fit, even with fewer data points.

Going beyond coarse averages to examining quality metrics for each layer weight matrix as a function of depth (or layer id), our metrics can be used to perform model diagnostics and to identify fine-scale properties in a pretrained model. Doing so involves separating $\hat{\alpha}$ into its two components, α and λ_{max}, and examining the distributions of each. We provide examples of this.

Layer Analysis: Metrics as a Function of Depth. Figure 4 plots the PL exponent α, as a function of depth, for each layer (first layer corresponds to data, last layer to labels) for the least accurate (shallowest) and most accurate (deepest) model in each of the VGG (no BN), ResNet, and DenseNet series. (Many more such plots are available at our repo.)

In the VGG models, Figure 4(a) shows that the PL exponent α systematically increases as we move down the network, from data to labels, in the Conv2D layers, starting with $\alpha \lesssim 2.0$

Series	#	Metric	$\langle \log \|\mathbf{W}\|_F^2 \rangle$	$\langle \log \|\mathbf{W}\|_\infty^2 \rangle$	$\hat{\alpha}$	$\langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle$
VGG	6	RMSE	0.56	**0.23**	0.48	0.34
		R^2	0.88	**0.98**	0.92	0.96
		Kendall-τ	-0.79	**-0.93**	**-0.93**	**-0.93**
ResNet	5	RMSE	0.9	0.97	**0.61**	0.66
		R^2	0.92	0.9	**0.96**	0.9
		Kendall-τ	-1.0	-1.0	-1.0	-1.0
ResNet-1K	19	RMSE	2.4	2.8	**1.8**	1.9
		R^2	0.81	0.74	**0.89**	0.88
		Kendall-τ	-0.79	-0.79	**-0.89**	-0.88
DenseNet	4	RMSE	0.3	**0.11**	0.16	0.21
		R^2	0.93	**0.99**	0.98	0.97
		Kendall-τ	-1.0	-1.0	-1.0	-1.0

Table 1: Quality metrics (RMSE, smaller is better; R^2, XXX is better; and Kendall-τ rank correlation, XXX is better) for reported Top1 test error for pretrained models in each architecture series. Column # refers to number of models. VGG, ResNet, and DenseNet were pretrained on ImageNet. ResNet-1K was pretrained on ImageNet-1K.

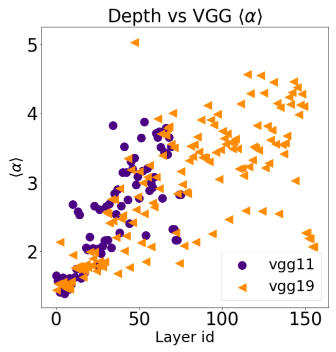
and reaching all the way to $\alpha \sim 5.0$; and then, in the last three, large, fully-connected (FC) layers, α stabilizes back down to $\alpha \in [2, 2.5]$. This is seen for all the VGG models (again, only the shallowest and deepest are shown), indicating that the main effect of increasing depth is to increase the range over which α increases, thus leading to larger α values in later Conv2D layers of the VGG models. This is quite different than the behavior of either the ResNet-1K models or the DenseNet models.

For the ResNet-1K models, Figure 4(b) shows that α also increases in the last few layers (more dramatically than for VGG, observe the differing scales on the Y axes). However, as the ResNet-1K models get deeper, there is a wide range over which α values tend to remain small. This is seen for other models in the ResNet-1K series, but it is most pronounced for the larger ResNet-1K (152) model, where α remains relatively stable at $\alpha \sim 2.0$, from the earliest layers all the way until we reach close to the final layers.

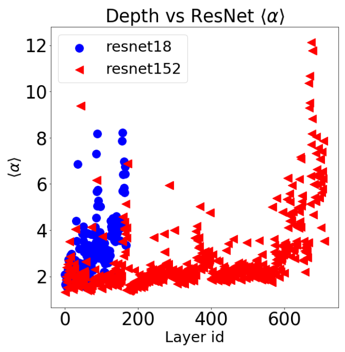
For the DenseNet models, Figure 4(c) shows that α tends to increase as the layer id increases, in particular for layers toward the end. While this is similar to the VGG models, with the DenseNet models, α values increase almost immediately after the first few layers, and the variance is much larger (in particular for the earlier and middle layers, where it can range all the way to $\alpha \sim 8.0$) and much less systematic throughout the network.

Overall, Figure 4 demonstrates that the distribution of α values among layers is architecture dependent, and that it can vary in a systematic way within an architecture series. This is to be expected, since some architectures enable better extraction of signal from the data. This also suggests that, while performing very well at predicting trends within an architecture series, PL-based metrics (as well as norm-based metrics) should be used with caution when comparing models with very different architectures.

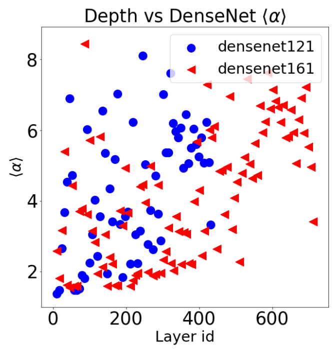
Correlation Flow; or How α Varies Across Layers. Figure 4 can be understood in terms of what we will call Correlation Flow. Recall that the average Log α-Norm metric and the Weighted Alpha metric are based on HT-SR Theory [1, 2, 3], which is in turn based on the statistical mechanics of heavy tailed and strongly correlated systems [21, 8, 22, 23]. There, one expects that the weight matrices of well-trained DNNs will exhibit correlations over many size scales, as is well-known in other strongly-correlated systems [21, 8]. This would imply that their



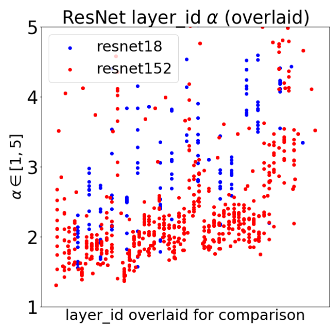
(a) VGG



(b) ResNet



(c) DenseNet



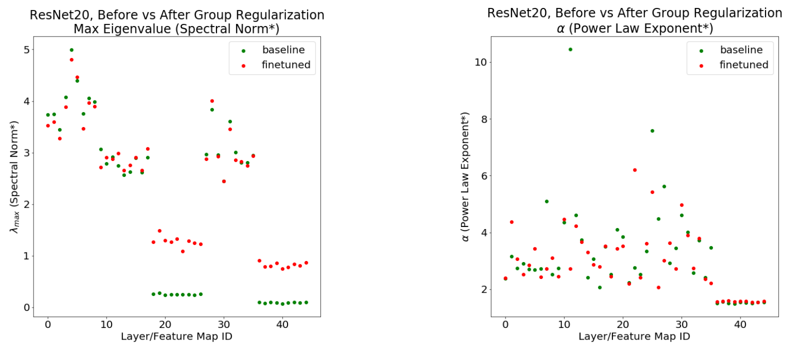
(d) ResNet (overlaid)

Figure 4: PL exponent (α) versus layer id, for the least and the most accurate models in VGG (a), ResNet (b), and DenseNet (c) series. (VGG is without BN; and note that the Y axes on each plot are different.) Subfigure (d) displays the ResNet models (b), zoomed in to $\alpha \in [1, 5]$, and with the layer ids overlaid on the X-axis, from smallest to largest, to allow a more detailed analysis of the most strongly correlated layers. Notice that ResNet152 exhibits different and much more stable behavior of α across layers. This contrasts with how both VGG models gradually worsen in deeper layers and how the DenseNet models are much more erratic. In the text, this is interpreted in terms of Correlation Flow.

ESDs can be well-fit by a truncated PL, with exponents $\alpha \in [2, 4]$. Much larger values ($\alpha \gg 6$) may reflect poorer PL fits, whereas smaller values ($\alpha \sim 2$), are associated with models that generalize better.

Informally, one would expect a DNN model to perform well when it facilitates the propagation of information/features across layers. In the absence of training/test data, one might hypothesize that this flow of information leaves empirical signatures on weight matrices, and that we can quantify this by measuring the PL properties of weight matrices. In this case, smaller α values correspond to layers in which information correlations between data across multiple scales are better captured [1, 8]. This leads to the hypothesis that small α values that are stable across multiple layers enable better correlation flow through the network. This is similar to recent work on the information bottleneck [24, 25], except that here we work in an entirely unsupervised setting.

Scale Collapse; or How Distillation May Break Models. The similarity between norm-based metrics and PL-based metrics may lead one to wonder whether the Weighted Alpha metric



(a) λ_{max} for ResNet20 layers (b) α for ResNet20 layers

Figure 5: ResNet20, distilled with Group Regularization, as implemented in the `distiller` (4D_regularized_5Lremoved) pretrained models. Log Spectral Norm ($\log \lambda_{max}$) and PL exponent (α) for individual layers, versus layer id, for both baseline (before distillation, green) and fine-tuned (after distillation, red) pretrained models.

is just a variation of more familiar norm-based metrics. Among hundreds of pretrained models, there are "exceptions that prove the rule," and these can be used to show that fitted α values do contain information not captured by norms. To illustrate this, we show that some compression/distillation methods [26] may actually damage models unexpectedly, by introducing what we call Scale Collapse, where several distilled layers have unexpectedly small Spectral Norms. By Scale Collapse, we mean that the size scale, e.g., as measured by the Spectral or Frobenius Norm, of one or more layers changes dramatically, while the size scale of other layers changes very little, as a function of some change to or perturbation of a model. The size scales of different parts of a DNN model are typically defined implicitly by the model training process, and they typically vary in a gradual way for high-quality models. Examples of changes of interest include model compression or distillation (discussed here for a CV model), data augmentation (discussed below for an NLP model), additional training, model fine-tuning, etc.

Consider ResNet20, trained on CIFAR10, before and after applying the Group Regularization distillation technique, as implemented in the `distiller` package [27]. We analyze the pretrained 4D_regularized_5Lremoved baseline and fine-tuned models. The reported baseline test accuracies (Top1= 91.45 and Top5= 99.75) are better than the reported fine-tuned test accuracies (Top1= 91.02 and Top5= 99.67). Because the baseline accuracy is greater, the previous results on ResNet (Table 1 and Figure 3) suggest that the baseline Spectral Norms should be smaller on average than the fine-tuned ones. The opposite is observed. Figure 5 presents the Spectral Norm (here denoted $\log \lambda_{max}$) and PL exponent (α) for each individual layer weight matrix $\mathbf{W}$. On the other hand, the α values (in Figure 5(b)) do not differ systematically between the baseline and fine-tuned models. Also, $\bar{\alpha}$, the average unweighted baseline α, from Eqn. (12), is smaller for the original model than for the fine-tuned model (as predicted by HT-SR Theory, the basis of $\hat{\alpha}$). In spite of this, Figure 5(b) also depicts two very large $\alpha \gg 6$ values for the baseline, but not for the fine-tuned, model. This suggests the baseline model has at least two over-parameterized/under-trained layers, and that the distillation method does, in fact, improve the fine-tuned model by compressing these layers.

Pretrained models in the `distiller` package have passed some quality metric, but they are much less well trodden than any of the VGG, ResNet, or DenseNet series. The obvious interpretation is that, while norms make good regularizers for a single model, there is no reason a priori to expect them correlate well with test accuracies across different models, and they may not differentiate well-trained versus poorly-trained models. We do expect, however, the PL α to do so, because it effectively measures the amount of information correlation in the model [1, 2, 3]. This suggests that the α values will improve, i.e., decrease, over time, as distillation techniques continue to improve. The reason for the anomalous behavior shown in Figure 5 is that the `distiller` Group Regularization technique causes the norms of the $\mathbf{W}$ pre-activation maps for two Conv2D layers to increase spuriously. This is difficult to diagnose by analyzing training/test curves, but it is easy to diagnose with our approach.

2.3 Comparison of NLP Models

Within the past few years, nearly 100 open source, pretrained NLP DNNs based on the revolutionary Transformer architecture have emerged. These include variants of BERT, Transformer-XML, GPT, etc. The Transformer architectures consist of blocks of so-called Attention layers, containing two large, Feed Forward (Linear) weight matrices [28]. In contrast to smaller pre-Activation maps arising in Cond2D layers, Attention matrices are significantly larger. In general, they have larger PL exponents α. Based on HT-SR Theory (in particular, the interpretation of values of $\alpha \sim 2$ as modeling systems with good correlations over many size scales [21, 8]), this suggests that these models fail to capture successfully many of the information correlations in the data (relative to their size) and thus are substantially under-trained. More generally, compared to CV models, modern NLP models have larger weight matrices and display different spectral properties.

While norm-based metrics perform reasonably well on well-trained NLP models, they often behave anomalously on poorly-trained models. For such models, weight matrices may display rank collapse, decreased Frobenius mass, or unusually small Spectral norms. This may be misinterpreted as “smaller is better.” Instead, it should probably be interpreted as being due to a similar mechanism to how distillation can “damage” otherwise good models. In contrast to norm-based metrics, PL-based metrics, including the Log α-Norm metric and the Weighted Alpha metric, display more consistent behavior, even on less well-trained models. To help identify when architectures need repair and when more and/or better data are needed, one can use these metrics, as well as the decomposition of the Weighted Alpha metric ($\alpha \log \lambda_{max}$) into its PL component (α) and its norm component ($\log \lambda_{max}$), for each layer.

Many NLP models, such as early variants of GPT and BERT, have weight matrices with unusually large PL exponents (e.g., $\alpha \gg 6$). This indicates these matrices may be under-correlated (i.e., over-parameterized, relative to the amount of data). In this regime, the truncated PL fit itself may not be very reliable because the Maximum Likelihood estimator it uses is unreliable in this range. In this case, the specific α values returned by the truncated PL fits are less reliable, but having large versus small α is reliable. If the ESD is visually examined, one can usually describe these $\mathbf{W}$ as in the Bulk-Decay or Bulk-plus-Spikes phase from HT-ST Theory [1, 2]. Previous work [1, 2] has conjectured that very well-trained DNNs would not have many outlier $\alpha \gg 6$. Consistent with this, more recent improved versions of GPT (shown below) and BERT (not shown) confirm this.

OpenAI GPT Models. The OpenAI GPT and GPT2 series of models provide the opportunity to analyze two effects: increasing the sizes of both the data set and the architectures simultaneously; and training the same model with low-quality data versus high-quality data. These models have the ability to generate fake text that appears to the human to be real, and they have generated media attention because of the potential for their misuse. For this reason, the original GPT model released by OpenAI was trained on a deficient data set, rendering the model interesting but not fully functional. Later, OpenAI released a much improved model, GPT2-small, which has

Series	#	$\langle \log \|\mathbf{W}\|_F \rangle$	$\langle \log \|\mathbf{W}\|_\infty \rangle$	$\hat{\alpha}$	$\langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle$
GPT	49	1.64	1.72	7.01	7.28
GPT2-small	49	2.04	2.54	9.62	9.87
GPT2-medium	98	2.08	2.58	9.74	10.01
GPT2-large	146	1.85	1.99	7.67	7.94
GPT2-xl	194	1.86	1.92	7.17	7.51

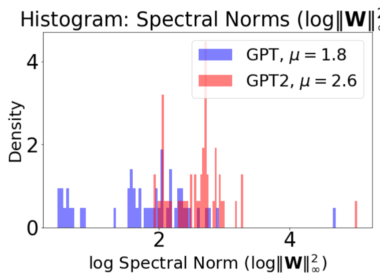
Table 2: Average value for the average Log Norm and Weighted Alpha metrics for pretrained OpenAI GPT and GPT2 models. Column # refers to number of layers treated. Averages do not include the first embedding layer(s) because they are not (implicitly) normalized. GPT has 12 layers, with 4 Multi-head Attention Blocks, giving 48 layer Weight Matrices, $\mathbf{W}$. Each Block has 2 components, the Self Attention (attn) and the Projection (proj) matrices. Self-attention matrices are larger, of dimension (2304×768) or (3072×768). The projection layer concatenates the self-attention results into a vector (of dimension 768). This gives 50 large matrices. Because GPT and GPT2 are trained on different data sets, the initial Embedding matrices differ in shape. GPT has an initial Token and Positional Embedding layers, of dimension (40478×768) and (512×768), respectively, whereas GPT2 has input Embeddings of shape (50257×768) and (1024×768), respectively. The OpenAI GPT2 (English) models are: GPT2-small, GPT2-medium, GPT2-large, and GPT2-xl, having 12, 24, 36, and 48 layers, respectively, with increasingly larger weight matrices.

the same architecture and number of layers as GPT, but which has been trained on a larger and better data set, making it remarkably good at generating (near) human-quality fake text. Subsequent models in the GPT2 were larger and trained to more data. By comparing GPT2-small to GPT2-medium to GPT2-large to GPT2-xl, we can examine the effect of increasing data set and model size simultaneously, as well as analyze well-trained versus very-well-trained models. By comparing the poorly-trained GPT to the well-trained GPT2-small, we can identify empirical indicators for when a model has been poorly-trained and thus may perform poorly when deployed. The GPT models we analyze are deployed with the popular HuggingFace PyTorch library [29].

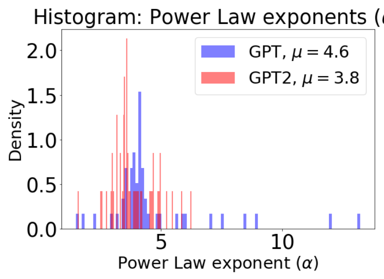
Average Quality Metrics for GPT and GPT2. We examine the performance of the four quality metrics (Log Frobenius norm, Log Spectral norm, Weighted Alpha, and Log α-Norm) for the OpenAI GPT and GPT2 pretrained models. See Table 2 for a summary of results. Comparing trends between GPT2-medium to GPT2-large to GPT2-xl, observe that (with one minor exception involving the log Frobenius norm metric) all four metrics decrease as one goes from medium to large to xl. This indicates that the larger models indeed look better than the smaller models, as expected. GPT2-small violates this general trend, but only very slightly. This could be due to under-optimization of the GPT2-small model, or since it is the smallest of the GPT2 series, and the metrics we present are most relevant for models at scale. Aside from this minor discrepancy, overall for these well-trained models, all these metrics now behave as expected, i.e., there is no Scale Collapse and norms are decreasing with increasing accuracy.

Comparing trends between GPT and GPT2-small reveals a different story. Observe that all four metrics increase when going from GPT to GPT2-small, i.e., they are larger for the higher-quality model (higher quality since GPT2-small was trained to better data) and smaller for the lower-quality model, when the number of layers is held fixed. This is unexpected. Here, too, we can perform model diagnostics, by separating $\hat{\alpha}$ into its two components, α and λ_{max}, and examining the distributions of each. In doing so, we see additional examples of Scale Collapse and additional evidence for Correlation Flow.

Layer Analysis: Scale Collapse in GPT and GPT2. We next examine the Spectral norm in GPT versus GPT2-small. In Figure 6(a), the poorly-trained GPT model has a smaller mean/median Spectral norm as well as, spuriously, many much smaller Spectral norms, compared to the well-trained GPT2-small. This violates the conventional wisdom that smaller Spectral norms are better. Because there are so many anomalously small Spectral norms, the GPT model appears to be exhibiting a kind of Scale Collapse, like that observed (in Figure 5) for the distilled CV models. This demonstrates that, while the Spectral (or Frobenius) norm may correlate well with predicted test error, at least among reasonably well-trained models, it is not a good indicator of the overall model quality in general. Naïvely using it as an empirical quality metric may give spurious results when applied to poorly-trained or otherwise deficient models.

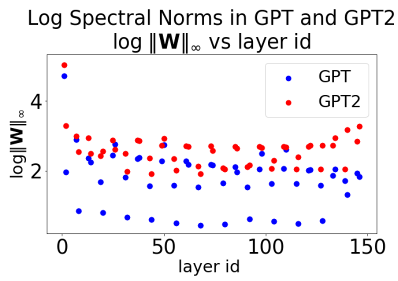


(a) Log Spectral Norm ($\log \|\mathbf{W}\|_\infty$)

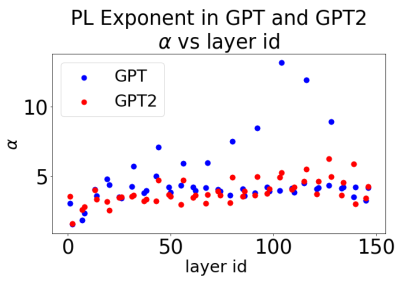


(b) PL exponent (α)

Figure 6: Histogram of PL exponents and Log Spectral Norms for weight matrices from the OpenAI GPT and GPT2-small pretrained models.



(a) Log Spectral Norm ($\log \|\mathbf{W}\|_\infty$)



(b) PL exponent (α)

Figure 7: Log Spectral Norms (in (a)) and PL exponents (in (b)) for weight matrices from the OpenAI GPT and GPT2-small pretrained models. (Note that the quantities shown on each Y axis are different.) In the text, this is interpreted in terms of Scale Collapse and Correlation Flow.

Figure 7(a) shows the Spectral norm as a function of depth (layer id). This illustrates two phenomenon. First, the large value of Spectral norm (in Figure 6(a)) corresponds to the first embedding layer(s). These layers have a different effective normalization, and therefore a different scale. See the Supplementary Information for details. We do not include them in our computed average metrics in Table 2. Second, for GPT, there seems to be two types of layers with very different Spectral norms (an effect which is seen, but to a much weaker extent, for GPT2-small). Recall that attention models have two types of layers, one small and large; and the Spectral norm (in particular, other norms do too) displays unusually small values for some of these layers for GPT. This Scale Collapse for the poorly-trained GPT is similar to what we observed for the distilled ResNet20 model in Figure 5(b). Because of the anomalous Scale Collapse that is frequently observed in poorly-trained models, these results suggest that scale-dependent norm metrics should not be directly applied to distinguish well-trained versus poorly-trained models.

Layer Analysis: Correlation Flow in GPT and GPT2. We next examine the distribution of α values in GPT versus GPT2-small. Figure 6(b) shows the histogram (empirical density), for all layers, of α for GPT and GPT2-small. The older deficient GPT has numerous unusually large α exponents—meaning they are not well-described by a PL fit. Indeed, we expect that a poorly-trained model will lack good (i.e., small α) PL behavior in many/most layers. On the other hand, the newer improved GPT2-small model has, on average, smaller α values than the older GPT, with all $\alpha \leq 6$ and with smaller mean/median α. It also has far fewer unusually-large outlying α values than GPT. From this (and other results not shown), we see that $\bar{\alpha}$ from Eqn. (12), provides a good quality metric for comparing the poorly-trained GPT versus the well-trained GPT2-small. This should be contrasted with the behavior displayed by scale-dependent metrics such as the Frobenius norm (not shown) and the Spectral norm. This also reveals why $\hat{\alpha}$ performs unusually in Table 2. The PL exponent α behaves as expected, and thus the scale-invariant $\bar{\alpha}$ metric lets us identify potentially poorly-trained models. It is the Scale Collapse that causes problems for $\hat{\alpha}$ (recall that the scale enters into $\hat{\alpha}$ via the weights $\log \lambda_{max}$).

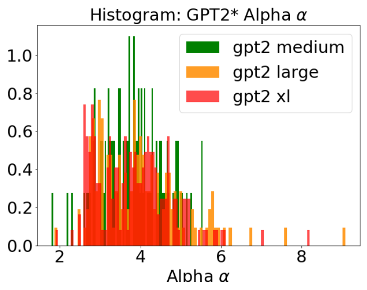
Figure 7(b) plots α versus the depth (layer id) for each model. The deficient GPT model displays two trends in α, one stable with $\alpha \sim 4$, and one increasing with layer id, with α reaching as high as 12. In contrast, the well-trained GPT2-small model shows consistent and stable patterns, again with one stable $\alpha \sim 3.5$ (and below the GPT trend), and the other only slightly trending up, with $\alpha \leq 6$. These results show that the behavior of α across layers differs significantly between GPT and GPT2-small, with the better GPT2-small looking more like the better ResNet-1K from Figure 4(b). These results also suggest that smaller more stable values of α across depth is beneficial, i.e., that the Correlation Flow is also a useful concept for NLP models.

GPT2: medium, large, xl. We now look across series of increasingly improving GPT2 models (well-trained versus very-well-trained models), by examining both the PL exponent α as well as the Log Norm metrics. Figure 8 shows the histograms over the layer weight matrices for fitted PL exponent α and the Log Alpha Norm metric. In general, and as expected, as we move from GPT2-medium to GPT2-xl, histograms for both α exponents and the Log Norm metrics downshift from larger to smaller values.

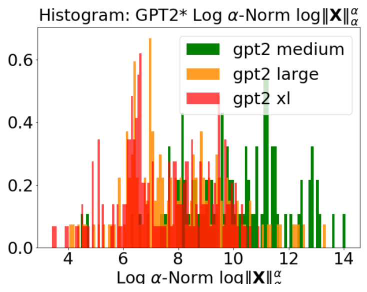
From Figure 8(a), we see that $\bar{\alpha}$, the average α value, decreases with increasing model size (XXX for GPT2-medium, XXX for GPT2-large, and XXX for GPT2-xl), although the differences are less noticeable between the differing well-trained versus very-well-trained GTP2 models than between the poorly-trained versus well-trained GPT and GPT2-small models. Also, from Figure 8(b), we see that, unlike GPT, the layer Log Alpha Norms behave more as expected for GPT2 layers, with the larger models consistently having smaller norms (XXX for GPT2-medium, XXX for GPT2-large, and XXX for GPT2-xl). Similarly, the Log Spectral Norm also decreases on average with the larger models (XXX for GPT2-medium, XXX for GPT2-large, and XXX for GPT2-xl). As expected, the norm metrics can indeed distinguish among well-trained versus very-well-trained models.

While the means and peaks of the α distributions are getting smaller, towards 2.0, as expected, Figure 8(a) also shows that the tails of the α distributions shift right, with larger GPT2 models having more unusually large α values. This is unexpected. It suggests that these larger GPT2 models are still under-optimized/over-parameterized (relative to the data on which they were trained) and that they have capacity to support datasets even larger than the recent XL

$1.5B$ release [30]. This does not contradict recent theoretical work on the benefits of over-parameterization [31], e.g., since in practice these extremely large models are not fully optimized. Subsequent refinements to these models, and other models such as BERT, indicate that this is likely the case.



(a) PL exponent (α)



(b) Log Alpha Norm

Figure 8: Histogram of PL exponents and Log Alpha Norm for weight matrices from models of different sizes in the GPT2 architecture series. (Plots omit the first 2 (embedding) layers, because they are normalized differently giving anomalously large values.)

2.4 Comparing Hundreds of Models

[michael: Refine text in this subsection after we get the final version of all the tables and figures in the appendix. Things to emphasize: we have a huge amount of data that we are summarizing very succicently; the table here is the tip of the iceberg; the results show our metrics are good, as a sanity check; they show that you can get some results in general, but looking at the details in the appendix show that results are more reliable within a given architecture series than across different architectures; point to out publicly-availab e repo; mention that we do Kendall-τ rank correlation also.]

We have performed a large-scale analysis of hundreds of publicly-available models. This broader analysis is on a much larger set of CV and NLP models, with a more diverse set of architectures, that have been developed for a wider range of tasks; and it complements the previous more detailed analysis on CV and NLP models, where we have analyzed only a single architecture series at a time. See the Supplementary Information (and our publicly-available repo) for details. To quantify the relationship between quality metrics and the reported test error and/or accuracy metrics, we use ordinary least squares (similar results are seen with rank correlation metrics) to regress the metrics on the Top1 (and Top5) reported errors (as dependent variables). These include Top5 errors for the ImageNet-1K model, percent error for the CIFAR-10/100, SVHN, CUB-200-2011 models, and Pixel accuracy (Pix.Acc.) and Intersection-Over-Union (IOU) for other models. We regress them individually on each of the norm-based and PL-based metrics.

Results are summarized in Table 10. For the mean, larger R^2 and smaller MSE are desirable; and for the standard deviation, smaller values are desirable. Taken as a whole, over the entire corpus of data, PL-based metrics are somewhat better for both the R^2 mean and standard deviation; and PL-based metrics are much better for MSE mean and standard deviation. These and other results suggest our conclusions hold much more generally.

	$\log \|\cdot\|_F$	$\log \|\cdot\|_\infty$	$\hat{\alpha}$	$\log \|\cdot\|_\alpha^\alpha$
R^2 (mean)	0.63	0.55	**0.64**	**0.64**
R^2 (std)	0.34	0.36	**0.29**	0.30
MSE (mean)	4.54	9.62	3.14	**2.92**
MSE (std)	8.69	23.06	5.14	**5.00**
Kendall-τ (mean)	XXX	XXX	XXX	XXX
Kendall-τ (std)	XXX	XXX	XXX	XXX

Table 3: Comparison of linear regression fits for different average Log Norm and Weighted Alpha metrics across 5 CV datasets, 17 architectures, covering 108 (out of over 400) different pretrained DNNs. We include regressions only for architectures with five or more data points, and which are positively correlated with test error. These results can be readily reproduced using the Google Colab notebooks. (See the Supplementary Information for details.) [charles: THIS TABLE IS AUTOGENERATED BUT I CNT INCLUDE FOR SOME REASON]

3 Discussion

Comparison of VGG, ResNet, and DenseNet Architectures. Going beyond the goal of predicting trends in the quality of state-of-the-art neural networks without access to training or testing data, observations such as the layer-wise observations we described in Figure 4 can be understood in terms of architectural differences between VGG, ResNet, and DenseNet. VGG resembles the traditional convolutional architectures, such as LeNet5, and consists of several [Conv2D-Maxpool-ReLu] blocks, followed by 3 large Fully Connected (FC) layers. ResNet greatly improved on VGG by replacing the large FC layers, shrinking the Conv2D blocks, and introducing residual connections. This optimized approach allows for greater accuracy with far fewer parameters, and ResNet models of up to 1000 layers have been trained [32].

The efficiency and effectiveness of ResNet seems to be reflected in the smaller and more stable $\alpha \sim 2.0$, across nearly all layers, indicating that the inner layers are very well correlated and more strongly optimized. This contrasts with the DenseNet models, which contains many connections between every layer. These results (large α, meaning that even a PL model is probably a poor fit) suggest that DenseNet has too many connections, diluting high quality interactions across layers, and leaving many layers very poorly optimized. Fine-scale measurements such as these enable us to form hypotheses as to the inner workings of DNN models, opening the door to an improved understanding of why DNNs work, as well as how to design better DNN models. Correlation Flow and Scale Collapse are two such examples.

Related work. Statistical mechanics has long had influence on DNN theory and practice [33, 34, 35]. Our best-performing PL-based metrics are based on statistical mechanics via HT-SR Theory [34, 1, 2, 36, 3]. The way in which we (and HT-SR Theory) use statistical mechanics theory is quite different than the way it is more commonly formulated [33, 35]. Going beyond idealized models, we use statistical mechanics in a broader sense, drawing upon techniques from quantitative finance, random matrix theory, and the statistical mechanics of heavy tailed and strongly correlated systems [21, 8, 22, 23]. There is also a large body of work in ML on using norm-based metrics to bound generalization error [37, 38, 39]. This theoretical work aims to prove generalization bounds, and this applied work then uses these norms to construct regularizers to improve training. Proving generalization bounds and developing new regularizers is very different than our focus of validating pretrained models.

Our work also has intriguing similarities and differences with work on understanding DNNs with the information bottleneck principle [24, 25], which posits that DNNs can be quantified by the mutual information between their layers and the input and output variables. Most importantly, our approach does not require access to any data, while information measures used in the information bottleneck approach do require this. Nevertheless, several results from HT-SR Theory, on which our metrics are based, have parallels in the information bottleneck approach. Perhaps most notably, the quick transition from a RANDOM-LIKE phase to BULK+SPIKES phase, followed by slow transition to a HEAVY-TAILED phase, as noted previously [1], is reminiscent of the dynamics on the Information Plane [25].

Finally, our work, starting in 2018 with the `WeightWatcher` tool [6], is the first to perform a detailed analysis of the weight matrices of DNNs [1, 2, 3]. Subsequent to the initial version of this paper, we became aware of two other works that were posted to the in 2020 within weeks of the initial version of this paper [40, 41]. Both of these papers validate our basic result that one can gain substantial insight into model quality by examining weight matrices without access to any training or testing data. However, both consider smaller models drawn from a much narrower range of applications than we consider. Previous results in HT-SR Theory suggest that insights from these smaller models may not extend to the state-of-the-art CV and NLP models we consider.

Conclusions. We have developed and evaluated methods to predict trends in the quality of state-of-the-art neural networks—without access to training or testing data. Our main methodology involved weight matrix meta-analysis, using the publicly-available `WeightWatcher` tool [6], and informed by the recently-developed HT-SR Theory [1, 2, 3]. Prior to our work, it was not even obvious that norm-based metrics would perform well to predict trends in quality across models (as they are usually used within a given model or parameterized model class, e.g., to bound generalization error or to construct regularizers). Our results are the first to demonstrate that they can be used for this important practical problem. Our results also demonstrate that PL-based metrics perform better than norm-based metrics. This should not be surprising—at least to those familiar with the statistical mechanics of heavy tailed and strongly correlated systems [21, 8, 22, 23]—since our use of PL exponents is designed to capture the idea that well-trained models capture information correlations over many size scales in the data. Again, though, our results are the first to demonstrate this. Our approach can also be used to provide fine-scale insight (rationalizing the flow of correlations or the collapse of size scale) throughout a network. Both Correlation Flow and Scale Collapse are important for improved diagnostics on pretrained models as well as for improved training methodologies.

Looking forward. More generally, our results suggest what a practical theory of DNNs should look like. To see this, let's distinguish between two types of theories: non-empirical or analogical theories, in which one creates, often from general principles, a very simple toy model that can be analyzed rigorously, and one then claims that the model is relevant to the system of interest; and semi-empirical theories, in which there exists a rigorous asymptotic theory, which comes with parameters, for the system of interest, and one then adjusts or fits those parameters to the finite non-asymptotic data, to make predictions about practical problems. A drawback of the former approach is that it typically makes very strong assumptions, and the strength of those assumptions can limit the practical applicability of the theory. Nearly all of the work on DNN theory focuses on the former type of theory. Our approach focuses on the latter type of theory. Our results, which are based on using sophisticated statistical mechanics theory and solving important practical DNN problems, suggests that the latter approach should be of interest more generally for those interested in developing a practical DNN theory.

4 Methods

To be fully reproducible, we only examine publicly-available, pretrained models. All of our computations were performed with the `WeightWatcher` tool (version 0.2.7) [6], and we provide all Jupyter and Google Colab notebooks used in an accompanying github repository [7], which includes more details and more results.

Additional Details on Layer Weight Matrices Recall that we can express the objective/optimization function for a typical DNN with L layers and with $N \times M$ weight matrices $\mathbf{W}_l$ and bias vectors $\mathbf{b}_l$ as Equation (2). We expect that most well-trained, production-quality models will employ one or more forms of regularization, such as Batch Normalization (BN), Dropout, etc., and many will also contain additional structure such as Skip Connections, etc. Here, we will ignore these details, and will focus only on the pretrained layer weight matrices $\mathbf{W}_l$. Typically, this model would be trained on some labeled data $\{d_i, y_i\} \in \mathcal{D}$, using Backprop, by minimizing the loss $\mathcal{L}$. For simplicity, we do not indicate the structural details of the layers (e.g., Dense or not, Convolutions or not, Residual/Skip Connections, etc.). Each layer is defined by one or more layer 2D weight matrices $\mathbf{W}_l$, and/or the 2D feature maps $\mathbf{W}_{l,i}$ extracted from 2D Convolutional (Conv2D) layers. A typical modern DNN may have anywhere between 5 and 5000 2D layer matrices.

For each Linear Layer, we get a single $(N \times M)$ (real-valued) 2D weight matrix, denoted $\mathbf{W}_l$, for layer l. This includes Dense or Fully-Connected (FC) layers, as well as 1D Convolutional (Conv1D) layers, Attention matrices, etc. We ignore the bias terms $\mathbf{b}_l$ in this analysis. Let the aspect ratio be $Q = \frac{N}{M}$, with $Q \geq 1$. For the Conv2D layers, we have a 4-index Tensor, of the form $(N \times M \times c \times d)$, consisting of $c \times d$ 2D feature maps of shape $(N \times M)$. We extract $n_l = c \times d$ 2D weight matrices $\mathbf{W}_{l,i}$, one for each feature map $i = [1, \ldots, n_l]$ for layer l.

SVD of Convolutional 2D Layers. There is some ambiguity in performing spectral analysis on Conv2D layers. Each layer is a 4-index tensor of dimension (w, h, in, out), with an $(w \times h)$ filter (or kernel) and (in, out) channels. When $w = h = k$, it gives $(k \times k)$ tensor slices, or pre-Activation Maps, $\mathbf{W}_{i,L}$ of dimension $(in \times out)$ each. We identify 3 different approaches for running SVD on a Conv2D layer:

1. run SVD on each pre-Activation Map $\mathbf{W}_{i,L}$, yielding $(k \times k)$ sets of M singular values;
2. stack the maps into a single matrix of, say, dimension $((k \times k \times out) \times in)$, and run SVD to get in singular values;
3. compute the 2D Fourier Transform (FFT) for each of the (in, out) pairs, and run SVD on the Fourier coefficients [42], leading to $\sim (k \times in \times out)$ non-zero singular values.

Each method has tradeoffs. Method (3) is mathematically sound, but computationally expensive. Method (2) is ambiguous. For our analysis, because we need thousands of runs, we select method (1), which is the fastest (and is easiest to reproduce).

Normalization of Empirical Matrices. Normalization is an important, if underappreciated, practical issue. Importantly, the normalization of weight matrices does *not* affect the PL fits because α is scale-invariant. Norm-based metrics, however, do depend strongly on the scale of the weight matrix—that is the point. To apply RMT, we usually define $\mathbf{X}$ with a $1/N$ normalization, assuming variance of $\sigma^2 = 1.0$. Pretrained DNNs are typically initialized with random weight matrices $\mathbf{W}_0$, with $\sigma^2 \sim 1/\sqrt{N}$, or some variant, e.g., the Glorot/Xavier normalization [43], or a $\sqrt{2/Nk^2}$ normalization for Convolutional 2D Layers. With this implicit scale, we do *not* "renormalize" the empirical weight matrices, i.e., we use them as-is. The only exception is that *we do rescale* the Conv2D pre-activation maps $\mathbf{W}_{i,L}$ by $k/\sqrt{2}$ so that they are on the same scale as the Linear / Fully Connected (FC) layers.

Special consideration for NLP models. NLP models, and other models with large initial embeddings, require special care because the embedding layers frequently lack the implicit $1/\sqrt{N}$ normalization present in other layers. For example, in GPT, for most layers, the maximum eigenvalue $\lambda_{max} \sim \mathcal{O}(10-100)$, but in the first embedding layer, the maximum eigenvalue is of order N (the number of words in the embedding), or $\lambda_{max} \sim \mathcal{O}(10^5)$. For GPT and GPT2, we treat all layers as-is (although one may want to normalize the first 2 layers $\mathbf{X}$ by $1/N$, or to treat them as outliers).

Acknowledgements. MWM would like to acknowledge ARO, DARPA, NSF, and ONR as well as the UC Berkeley BDD project and a gift from Intel for providing partial support of this work. Our conclusions do not necessarily reflect the position or the policy of our sponsors, and no official endorsement should be inferred. We would also like to thank Amir Khosrowshahi and colleagues at Intel for helpful discussion regarding the Group Regularization distillation technique.

References

[1] C. H. Martin and M. W. Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. Technical Report Preprint: arXiv:1810.01075, 2018.

[2] C. H. Martin and M. W. Mahoney. Traditional and heavy-tailed self regularization in neural network models. In *Proceedings of the 36th International Conference on Machine Learning*, pages 4284–4293, 2019.

[3] C. H. Martin and M. W. Mahoney. Heavy-tailed Universality predicts trends in test accuracies for very large pre-trained deep neural networks. In *Proceedings of the 20th SIAM International Conference on Data Mining*, 2020.

[4] K. Janocha and W. M. Czarnecki. On loss functions for deep neural networks in classification. Technical Report Preprint: arXiv:1702.05659, 2017.

[5] Sandbox for training convolutional networks for computer vision. https://github.com/osmr/imgclsmob.

[6] WeightWatcher, 2018. https://pypi.org/project/WeightWatcher/.

[7] https://github.com/CalculatedContent/ww-trends-2020.

[8] D. Sornette. *Critical phenomena in natural sciences: chaos, fractals, selforganization and disorder: concepts and tools.* Springer-Verlag, Berlin, 2006.

[9] H. Nishimori. *Statistical Physics of Spin Glasses and Information Processing: An Introduction.* Oxford University Press, Oxford, 2001.

[10] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: an explanation of $1/f$ noise. *Physical Review Letters*, 59(4):381–384, 1987.

[11] N. W. Watkins, G. Pruessner, S. C. Chapman, N. B. Crosby, and H. J. Jensen. 25 years of self-organized criticality: Concepts and controversies. *Space Science Reviews*, 198:3–44, 2016.

[12] L. Hodgkinson and M. W. Mahoney. Multiplicative noise and heavy tails in stochastic optimization,. Technical Report Preprint: arXiv:2006.06293, 2020.

[13] D. Sornette and R. Cont. Convergent multiplicative processes repelled from zero: Power laws and truncated power laws. *Journal De Physique I*, 7:431–444, 1997.

[14] W. L. Shew, H. Yang, S. Yu, R. Roy, and D. Plenz. Information capacity and transmission are maximized in balanced cortical networks with neuronal avalanches. *The Journal of Neuroscience*, 31(1):55–63, 2011.

[15] S. Yu, A. Klaus, H. Yang, and D. Plenz. Scale-invariant neuronal avalanche dynamics and the cut-off in size distributions. *PLoS ONE*, 9(6):e99761, 2014.

[16] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[17] J. Alstott, E. Bullmore, and D. Plenz. powerlaw: A python package for analysis of heavy-tailed distributions. *PLoS ONE*, 9(1):e85777, 2014.

[18] M. E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46:323–351, 2005.

[19] O. Russakovsky et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[20] A. Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Annual Advances in Neural Information Processing Systems 32: Proceedings of the 2019 Conference*, pages 8024–8035, 2019.

[21] J. P. Bouchaud and M. Potters. *Theory of Financial Risk and Derivative Pricing: From Statistical Physics to Risk Management*. Cambridge University Press, 2003.

[22] J. P. Bouchaud and M. Potters. Financial applications of random matrix theory: a short review. In G. Akemann, J. Baik, and P. Di Francesco, editors, *The Oxford Handbook of Random Matrix Theory*. Oxford University Press, 2011.

[23] J. Bun, J.-P. Bouchaud, and M. Potters. Cleaning large correlation matrices: tools from random matrix theory. *Physics Reports*, 666:1–109, 2017.

[24] N. Tishby and N. Zaslavsky. Deep learning and the information bottleneck principle. In *Proceedings of the 2015 IEEE Information Theory Workshop, ITW 2015*, pages 1–5, 2015.

[25] R. Shwartz-Ziv and N. Tishby. Opening the black box of deep neural networks via information. Technical Report Preprint: arXiv:1703.00810, 2017.

[26] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. A survey of model compression and acceleration for deep neural networks. Technical Report Preprint: arXiv:1710.09282, 2017.

[27] Intel Distiller package. https://nervanasystems.github.io/distiller.

[28] A. Vaswani et al. Attention is all you need. Technical Report Preprint: arXiv:1706.03762, 2017.

[29] T. Wolf et al. Huggingface's transformers: State-of-the-art natural language processing. Technical Report Preprint: arXiv:1910.03771, 2019.

[30] OpenAI GPT-2: 1.5B Release. https://openai.com/blog/gpt-2-1-5b-release/.

[31] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proc. Natl. Acad. Sci. USA*, 116:15849–15854, 2019.

[32] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. Technical Report Preprint: arXiv:1603.05027, 2016.

[33] A. Engel and C. P. L. Van den Broeck. *Statistical mechanics of learning*. Cambridge University Press, New York, NY, USA, 2001.

[34] C. H. Martin and M. W. Mahoney. Rethinking generalization requires revisiting old ideas: statistical mechanics approaches and complex learning behavior. Technical Report Preprint: arXiv:1710.09553, 2017.

[35] Y. Bahri, J. Kadmon, J. Pennington, S. Schoenholz, J. Sohl-Dickstein, and S. Ganguli. Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*, 11:501–528, 2020.

[36] C. H. Martin and M. W. Mahoney. Statistical mechanics methods for discovering knowledge from modern production quality neural networks. In *Proceedings of the 25th Annual ACM SIGKDD Conference*, pages 3239–3240, 2019.

[37] B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In *Proceedings of the 28th Annual Conference on Learning Theory*, pages 1376–1401, 2015.

[38] P. Bartlett, D. J. Foster, and M. Telgarsky. Spectrally-normalized margin bounds for neural networks. Technical Report Preprint: arXiv:1706.08498, 2017.

[39] Q. Liao, B. Miranda, A. Banburski, J. Hidary, and T. Poggio. A surprising linear relationship predicts test performance in deep networks. Technical Report Preprint: arXiv:1807.09659, 2018.

[40] G. Eilertsen, D. Jönsson, T. Ropinski, J. Unger, and A. Ynnerman. Classifying the classifier: dissecting the weight space of neural networks. Technical Report Preprint: arXiv:2002.05688, 2020.

[41] T. Unterthiner, D. Keysers, S. Gelly, O. Bousquet, and I. Tolstikhin. Predicting neural network accuracy from weights. Technical Report Preprint: arXiv:2002.11448, 2020.

[42] H. Sedghi, V. Gupta, and P. M. Long. The singular values of convolutional layers. Technical Report Preprint: arXiv:1805.10408, 2018.

[43] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Workshop on Artificial Intelligence and Statistics*, pages 249–256, 2010.

A Supplementary Information

A.1 Supplementary Details

Reproducing Sections 2.2 and 2.3. We provide a github repository for this paper that includes Jupyter notebooks that fully reproduce all results (as well as many other results) [7]. All results have been produced using the `WeightWatcher` tool (v0.2.7) [6]. The ImageNet and OpenAI GPT pretrained models are provided in the current pyTorch [20] and Huggingface [29] distributions, as specified in the `requirements.txt` file.

Reproducing Figure 5, for the Distiller Model. In the `distiller` folder of our github repo, we provide the original Jupyter Notebooks, which use the Intel `distiller` framework [27]. Figure 5 is from the `‘‘...-Distiller-ResNet20.ipynb’’` notebook (see Table 4). For completeness, we provide both the results described here, as well as additional results on other pretrained and distilled models using the `WeightWatcher` tool.

Reproducing Table 10 in Section 2.4. In the `ww-colab` folder of our github repo, we provide several Google Colab notebooks which can be used to reproduce the results of Section 2.4. The ImageNet-1K and other pretrained models are taken from the pyTorch models in the `omsr/imgclsmob` "Sandbox for training convolutional networks for computer vision" github repository [5]. The data for each regression can be generated in parallel by running each Google Colab notebook (i.e., `ww_colab_0_100.ipynb`) simultaneously on the same account. The data generated are analyzed with `ww_colab_results.ipynb`, which runs all regressions and which tabulates the results presented in Table 10.

We attempt to run linear regressions for all pyTorch models for each architecture series for all datasets provided. There are over 450 models in all, and we note that the `osmr/imgclsmob`

Table	Figure	Jupyter Notebook
	2	WeightWatcher-VGG.ipynb
	3(a)	WeightWatcher-ResNet.ipynb
	3(b)	WeightWatcher-ResNet-1K.ipynb
	4(a)	WeightWatcher-VGG.ipynb
	4(b)	WeightWatcher-ResNet.ipynb
	4(c)	WeightWatcher-DenseNet.ipynb
	5	WeightWatcher-Intel-Distiller-ResNet20.ipynb
	6	WeightWatcher-OpenAI-GPT.ipynb
	7, 8	WeightWatcher-OpenAI-GPT2.ipynb
3,7,8,9	Appendix	OSMR-Analysis.ipynb

Table 4: Jupyter notebooks used to reproduce all results in Sections 2.2 and 2.3.[charles: Need to update referendes]

Dataset	# of Models
imagenet-1k	76
svhn	30
cifar-100	26
cifar-10	18
cub-200-2011	12

Table 5: Datasets used [charles: This table needs updated]

repository is constantly being updated with new models. We omit the results for CUB-200-2011, Pascal-VOC2012, ADE20K, and COCO datasets, as there are fewer than 15 models for those datasets. Also, we filter out regressions with fewer than 5 datapoints.

We remove the following outliers, as identified by visual inspection: `efficient_b0,_b2`. We also remove the entire `cifar100 ResNeXT` series, which is the only example to show no trends with the norm metrics. The final datasets used are shown in Table 5. The final architecture series used are shown in Table 6, with the number of models in each.

Tables and figures summarizing this analysis (in a more fine-grained way than provided by Table 10) are presented next.

A.2 Supplementary Tables and Figures

[michael: The text in this section needs to be changed—probably not removed, but instead discuss things, not point to the figure we remove, and point to the repo.]

[michael: Do we get more visually compelling results if we present figures, but segmented by architectures, as listed in Table 6; if so, perhaps we should put those, so we don't tell R1 that we just removed what looked bad; or maybe actually both, since we segment by dataset in Table 5.]

To explain further how to reproduce our analysis, we run three batches of linear regressions. First, at the global level, we divide models by datasets and run regressions separately on all models of a certain dataset, regardless of the architecture. At this level, the plots are quite noisy and clustered, as each architecture has its own accuracy trend; but one can still see that most plots show positive relationship with positive coefficients. [michael: Comment here on differences.]

To generate the results in Table 10, we run linear regressions for each architecture series in Table 6, regressing each empirical Log Norm metric against the reported Top1 (and Top5) errors (as listed on the `osmr/imgclsmob` github repository README file [5], with the relevant

Architecture	# of Models	Datasets				
	total	imagenet-1k	cifar-10	cifar-100	svhn	cub-200-2011
ResNet	51	22	8	8	7	6
EfficientNet	20	20	0	0	0	0
PreResNet	14	14	0	0	0	0
VGG/BN-VGG	12	12	0	0	0	0
ShuffleNet	12	12	0	0	0	0
DLA	10	10	0	0	0	0
HRNet	9	9	0	0	0	0
DRN-C/DRN-D	7	7	0	0	0	0
SqueezeNext/SqNxt	6	6	0	0	0	0
ESPNetv2	5	5	0	0	0	0
SqueezeNet/SqueezeResNet	4	4	0	0	0	0
IGCV3	4	4	0	0	0	0
ProxylessNAS	4	4	0	0	0	0
DIA-ResNet/DIA-PreResNet	24	0	8	8	8	0
SENet/SE-ResNet	20	0	5	5	4	6
WRN	8	0	0	4	4	0
ResNeXt	4	0	0	0	4	0

Table 6: Architectures used in total and per dataset

data extracted and provided in our github repo as `pytorchcv.html`). We record the R^2 and MSE for each metric, averaged over all regressions for all architectures and datasets. [michael: Mention KTau.] See Table ??, Table 8, and Table ??. In the repo, plots are provided for every regression, and more fine grained results may be computed by the reader by analyzing the data in the `df_all.xlsx` file. The final analysis includes 108 regressions in all, those with 4 or more models, with a positive R^2.

See Figure 9, Figure 10, Figure 11, Figure 12, Figure 13, Figure 14, Figure 15, Figure 16, and Figure 9 for more details. [michael: Put a bit of discussion, e.g., on best and worst.]

[michael: Probably some discussion of why these metrics are the right ones, e.g., as opposed to p values, to address R1 comments.]

A.3 Supplementary Discussion: Additional Details on HT-SR Theory

The original work on HT-SR Theory [1, 2, 3] considered NNs including AlexNet and InceptionV3 (as well as DenseNet, ResNet, and VGG), and it showed that for nearly every $\mathbf{W}$, the (bulk and tail) of the ESDs can be fit to a truncated PL and the PL exponents α nearly all lie within the range $\alpha \in (1.5, 5)$. Our meta-analysis, the main results of which are summarized in this paper, has shown that these results are ubiquitous. For example, upon examining nearly 10,000 layer weight matrices $\mathbf{W}_{l,i}$ across hundreds of different modern pre-trained DNN architectures, the ESD of nearly every $\mathbf{W}$ layer matrix can be fit to a truncated PL: $70-80\%$ of the time, the fitted PL exponent α lies in the range $\alpha \in (2, 4)$; and $10-20\%$ of the time, the fitted PL exponent α lies in the range $\alpha < 2$. Of course, there are exceptions: in any real DNN, the fitted α may range anywhere from ~ 1.5 to 10 or higher (and, of course, larger values of α may indicate that the PL is not a good model for the data). Still, overall, in nearly all large, pre-trained DNNs, the correlations in the weight matrices exhibit a remarkable Universality, being both Heavy Tailed, and having small—but not too small—PL exponents.

Dataset	Model	$\langle\log\|\cdot\|_F^2\rangle$	$\langle\log\|\cdot\|_\infty^2\rangle$	$\hat{\alpha}$	$\langle\log\|\cdot\|_\alpha^\alpha\rangle$
imagenet-1k	ResNet	2.52	3.35	**1.91**	2.04
imagenet-1k	EfficientNet	1.64	**1.11**	1.60	1.58
imagenet-1k	PreResNet	2.57	3.93	**1.90**	1.93
imagenet-1k	VGG	0.92	**0.83**	1.37	1.26
imagenet-1k	ShuffleNet	5.95	9.46	4.42	**4.30**
imagenet-1k	DLA	4.79	**3.02**	3.94	4.06
imagenet-1k	HRNet	0.68	0.72	**0.40**	0.40
imagenet-1k	DRN-C	0.77	0.81	**0.64**	0.69
imagenet-1k	SqueezeNext	4.68	4.62	3.65	**3.64**
imagenet-1k	ESPNetv2	3.71	3.84	**1.37**	1.59
imagenet-1k	SqueezeNet	0.33	0.33	**0.26**	0.29
imagenet-1k	IGCV3	1.39	9.37	2.91	**1.04**
imagenet-1k	ProxylessNAS	**0.44**	0.51	0.53	0.51
cifar-10	ResNet	0.56	0.55	**0.53**	0.53
cifar-10	DIA-ResNet	**0.22**	0.28	0.53	0.56
cifar-10	SENet	0.30	0.30	**0.20**	0.20
cifar-100	ResNet	2.03	2.12	**1.75**	1.75
cifar-100	DIA-ResNet	**0.60**	1.17	0.96	1.01
cifar-100	SENet	0.60	0.65	**0.51**	0.51
cifar-100	WRN	0.37	0.44	0.26	**0.25**
svhn	ResNet	0.20	0.20	**0.15**	0.16
svhn	DIA-ResNet	0.07	**0.06**	0.13	0.13
svhn	SENet	**0.04**	0.07	0.05	0.05
svhn	WRN	0.07	0.08	0.07	**0.07**
svhn	ResNeXt	0.06	**0.06**	0.11	0.09
cub-200-2011	ResNet	0.45	**0.42**	1.79	1.79
cub-200-2011	SENet	**1.03**	1.13	1.36	1.40

Table 7: RMSE Results for our analysis of all CV models in Table 5.

Dataset	Model	$\langle\log\|\cdot\|_F^2\rangle$	$\langle\log\|\cdot\|_\infty^2\rangle$	$\hat{\alpha}$	$\langle\log\|\cdot\|_\alpha^\alpha\rangle$
imagenet-1k	ResNet	0.79	0.63	**0.88**	0.86
imagenet-1k	EfficientNet	0.65	**0.84**	0.67	0.67
imagenet-1k	PreResNet	0.73	0.36	**0.85**	0.85
imagenet-1k	VGG	0.71	**0.76**	0.35	0.46
imagenet-1k	ShuffleNet	0.63	0.06	0.80	**0.81**
imagenet-1k	DLA	0.11	**0.65**	0.40	0.36
imagenet-1k	HRNet	0.91	0.90	**0.97**	0.97
imagenet-1k	DRN-C	0.81	0.79	**0.87**	0.85
imagenet-1k	SqueezeNext	0.05	0.07	0.42	**0.43**
imagenet-1k	ESPNetv2	0.42	0.38	**0.92**	0.89
imagenet-1k	SqueezeNet	0.01	0.00	**0.38**	0.26
imagenet-1k	IGCV3	0.98	0.12	0.92	**0.99**
imagenet-1k	ProxylessNAS	**0.68**	0.56	0.53	0.58
cifar-10	ResNet	0.58	0.59	**0.62**	0.61
cifar-10	DIA-ResNet	**0.96**	0.93	0.74	0.71
cifar-10	SENet	0.91	0.91	**0.96**	0.96
cifar-100	ResNet	0.61	0.58	**0.71**	0.71
cifar-100	DIA-ResNet	**0.96**	0.85	0.90	0.89
cifar-100	SENet	0.97	0.96	**0.98**	0.98
cifar-100	WRN	0.32	0.04	0.66	**0.69**
svhn	ResNet	0.69	0.70	**0.82**	0.81
svhn	DIA-ResNet	0.94	**0.95**	0.78	0.77
svhn	SENet	**0.99**	0.96	0.98	0.98
svhn	WRN	0.13	0.10	0.20	**0.21**
svhn	ResNeXt	0.87	**0.90**	0.64	0.75
cub-200-2011	ResNet	0.94	**0.95**	0.08	0.08
cub-200-2011	SENet	**0.66**	0.59	0.41	0.38

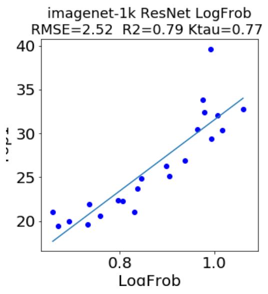
Table 8: R2 Results for our analysis of all CV models in Table 5.

Dataset	Model	$\langle \log \|\cdot\|_F^2 \rangle$	$\langle \log \|\cdot\|_\infty^2 \rangle$	$\hat{\alpha}$	$\langle \log \|\cdot\|_\alpha^\alpha \rangle$
imagenet-1k	ResNet	0.77	0.74	**0.88**	0.88
imagenet-1k	EfficientNet	0.67	**0.79**	0.66	0.66
imagenet-1k	PreResNet	0.65	0.54	**0.87**	0.87
imagenet-1k	VGG	**0.82**	0.76	0.52	0.61
imagenet-1k	ShuffleNet	0.39	0.09	**0.85**	0.82
imagenet-1k	DLA	0.51	**0.82**	0.78	0.69
imagenet-1k	HRNet	0.56	0.44	**0.61**	0.61
imagenet-1k	DRN-C	**0.90**	0.81	0.90	0.90
imagenet-1k	SqueezeNext	0.47	-0.33	**0.73**	0.73
imagenet-1k	ESPNetv2	0.00	0.80	**1.00**	1.00
imagenet-1k	SqueezeNet	0.33	0.00	**0.67**	0.67
imagenet-1k	IGCV3	**1.00**	0.00	1.00	1.00
imagenet-1k	ProxylessNAS	**0.33**	0.33	0.33	0.33
cifar-10	ResNet	0.64	0.64	**0.71**	0.71
cifar-10	DIA-ResNet	0.79	0.50	**0.86**	0.79
cifar-10	SENet	0.74	**0.95**	0.95	0.95
cifar-100	ResNet	**0.64**	0.57	0.64	0.64
cifar-100	DIA-ResNet	**0.93**	0.43	0.93	0.93
cifar-100	SENet	**1.00**	1.00	1.00	1.00
cifar-100	WRN	**0.67**	-0.67	0.00	0.00
svhn	ResNet	**0.81**	0.71	0.81	0.81
svhn	DIA-ResNet	**0.86**	0.86	0.57	0.57
svhn	SENet	**1.00**	1.00	0.67	0.67
svhn	WRN	-0.33	-0.33	**0.67**	0.67
svhn	ResNeXt	**0.67**	0.67	0.33	0.33
cub-200-2011	ResNet	**1.00**	1.00	-0.33	-0.33
cub-200-2011	SENet	**0.87**	0.87	-0.20	-0.20

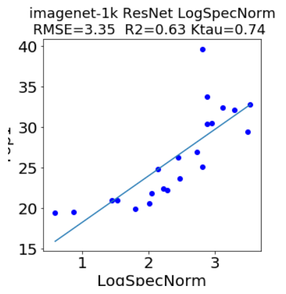
Table 9: Ktau Results for our analysis of all CV models in Table 5.

	$\log \|\cdot\|_F^2$	$\log \|\cdot\|_\infty^2$	$\hat{\alpha}$	$\log \|\cdot\|_\alpha^\alpha$
RMSE (mean)	4.84	5.56	4.58	4.54
RMSE (std)	9.14	9.17	9.16	9.17
R2 (mean)	3.9	3.85	3.89	3.89
R2 (std)	9.34	9.36	9.34	9.34
R2 adjusted (mean)	3.82	3.77	3.82	3.83
R2 adjusted (std)	9.38	9.4	9.37	9.37
Kendal-tau (mean)	3.84	3.77	3.85	3.85
Kendal-tau (std)	9.37	9.4	9.36	9.36

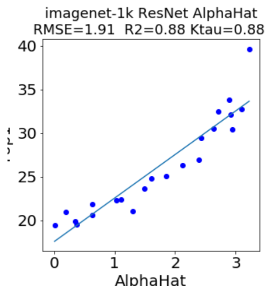
Table 10: Comparison of linear regression fits for different average Log Norm and Weighted Alpha metrics across 5 CV datasets, 17 architectures, covering 108 (out of over 400) different pretrained DNNs. We include regressions only for architectures with five or more data points, and which are positively correlated with test error. These results can be readily reproduced using the Google Colab notebooks. (See the Supplementary Information for details.)



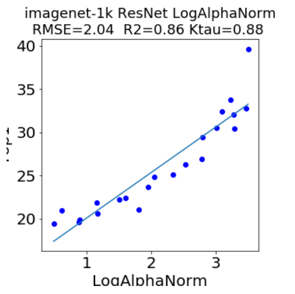
(a) imagenet-1k–ResNet, $\langle \log \|\cdot\|_F \rangle$



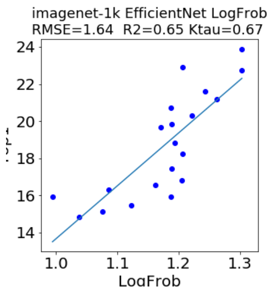
(b) imagenet-1k–ResNet, $\langle \log \|\cdot\|_\infty \rangle$



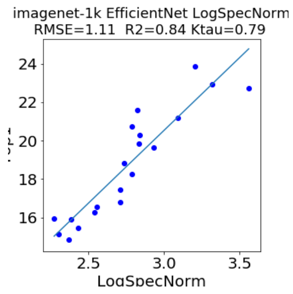
(c) imagenet-1k–ResNet, $\hat{\alpha}$



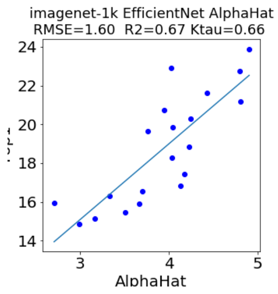
(d) imagenet-1k–ResNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



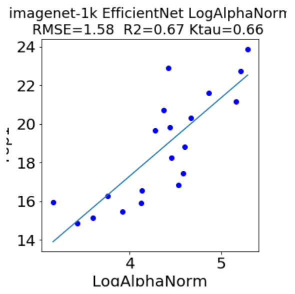
(e) imagenet-1k–EfficientNet, $\langle \log \|\cdot\|_F \rangle$



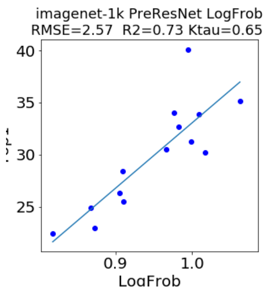
(f) imagenet-1k–EfficientNet, $\langle \log \|\cdot\|_\infty \rangle$



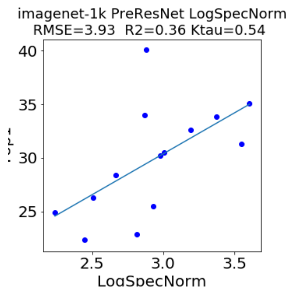
(g) imagenet-1k–EfficientNet, $\hat{\alpha}$



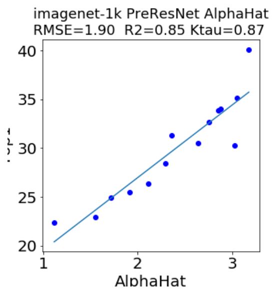
(h) imagenet-1k–EfficientNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



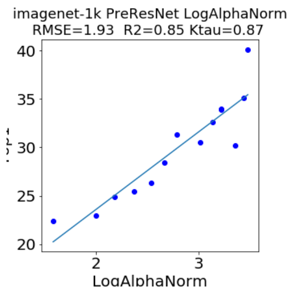
(i) imagenet-1k–PreResNet, $\langle \log \|\cdot\|_F \rangle$



(j) imagenet-1k–PreResNet, $\langle \log \|\cdot\|_\infty \rangle$

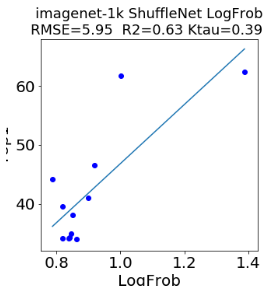


(k) imagenet-1k–PreResNet, $\hat{\alpha}$

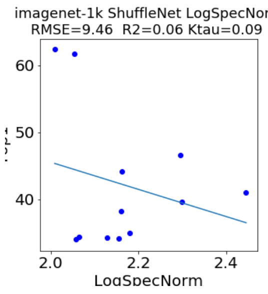


(l) imagenet-1k–PreResNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$

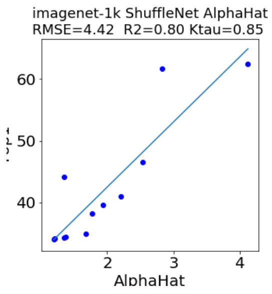
Figure 9: Regression plots for model-dataset pairs, based on data from Table 7, Table 8, and Table 9. Each row corresponds to a different dataset-model pair: imagenet-1k–ResNet; imagenet-1k–EfficientNet; and imagenet-1k–PreResNet; repsectively. Each column corresponds to a different metric: $\langle \log \|\cdot\|_F \rangle$; $\langle \log \|\cdot\|_\infty \rangle$; $\hat{\alpha}$; and $\langle \log \|\cdot\|_\alpha^\alpha \rangle$; repsectively.



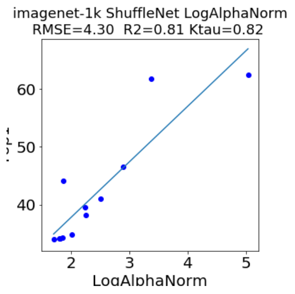
(a) imagenet-1k–ShuffleNet, $\langle \log \|\cdot\|_F \rangle$



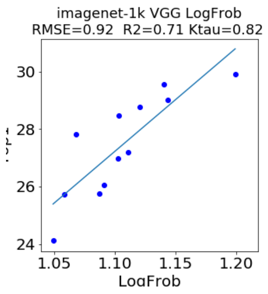
(b) imagenet-1k–ShuffleNet, $\langle \log \|\cdot\|_\infty \rangle$



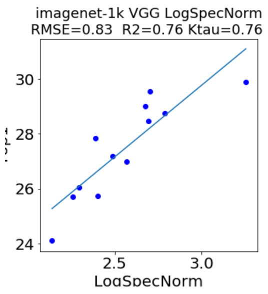
(c) imagenet-1k–ShuffleNet, $\hat{\alpha}$



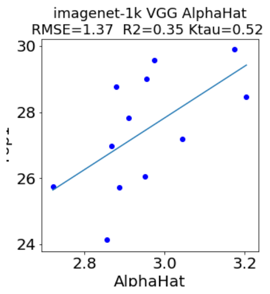
(d) imagenet-1k–ShuffleNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



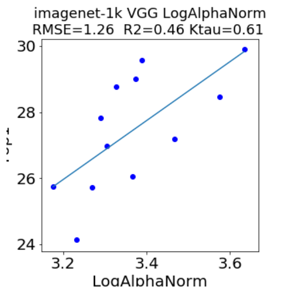
(e) imagenet-1k–VGG, $\langle \log \|\cdot\|_F \rangle$



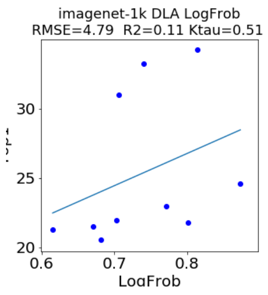
(f) imagenet-1k–VGG, $\langle \log \|\cdot\|_\infty \rangle$



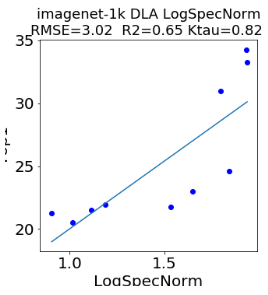
(g) imagenet-1k–VGG, $\hat{\alpha}$



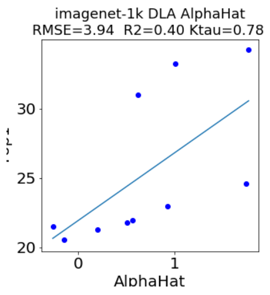
(h) imagenet-1k–VGG, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



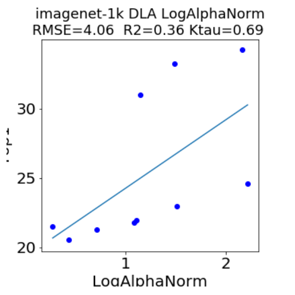
(i) imagenet-1k–DLA, $\langle \log \|\cdot\|_F \rangle$



(j) imagenet-1k–DLA, $\langle \log \|\cdot\|_\infty \rangle$

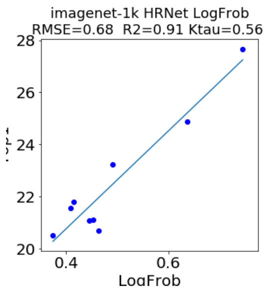


(k) imagenet-1k–DLA, $\hat{\alpha}$

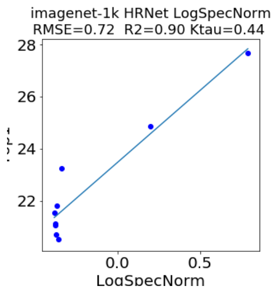


(l) imagenet-1k–DLA, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$

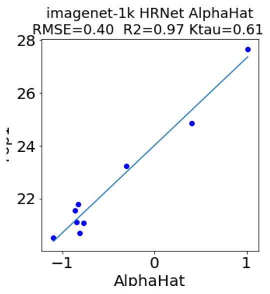
Figure 10: Regression plots for model-dataset pairs, based on data from Table 7, Table 8, and Table 9. Each row corresponds to a different dataset-model pair: imagenet-1k–ShuffleNet; imagenet-1k–VGG; and imagenet-1k–DLA; repsectively. Each column corresponds to a different metric: $\langle \log \|\cdot\|_F \rangle$; $\langle \log \|\cdot\|_\infty \rangle$; $\hat{\alpha}$; and $\langle \log \|\cdot\|_\alpha^\alpha \rangle$; repsectively.



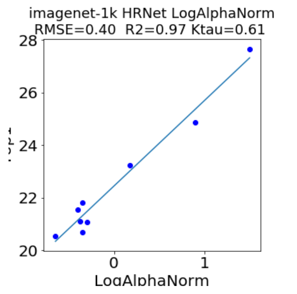
(a) imagenet-1k–HRNet, $\langle \log \| \cdot \|_F \rangle$



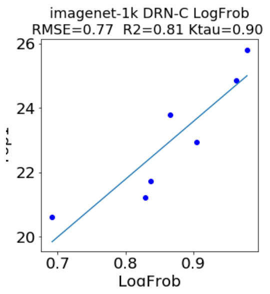
(b) imagenet-1k–HRNet, $\langle \log \| \cdot \|_\infty \rangle$



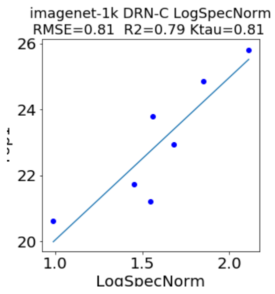
(c) imagenet-1k–HRNet, $\hat{\alpha}$



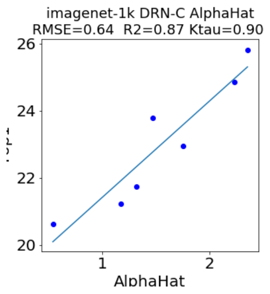
(d) imagenet-1k–HRNet, $\langle \log \| \cdot \|_\alpha^\alpha \rangle$



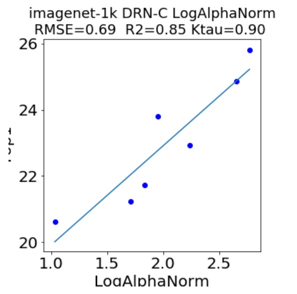
(e) imagenet-1k–DRN-C, $\langle \log \| \cdot \|_F \rangle$



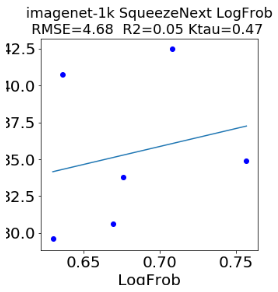
(f) imagenet-1k–DRN-C, $\langle \log \| \cdot \|_\infty \rangle$



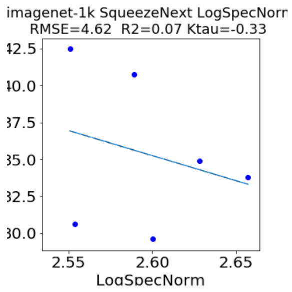
(g) imagenet-1k–DRN-C, $\hat{\alpha}$



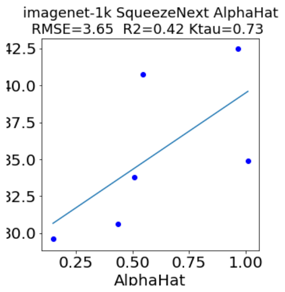
(h) imagenet-1k–DRN-C, $\langle \log \| \cdot \|_\alpha^\alpha \rangle$



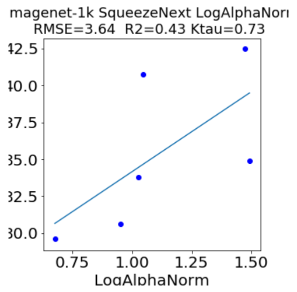
(i) imagenet-1k–SqueezeNext, $\langle \log \| \cdot \|_F \rangle$



(j) imagenet-1k–SqueezeNext, $\langle \log \| \cdot \|_\infty \rangle$

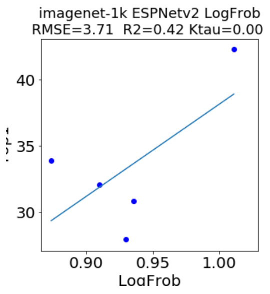


(k) imagenet-1k–SqueezeNext, $\hat{\alpha}$

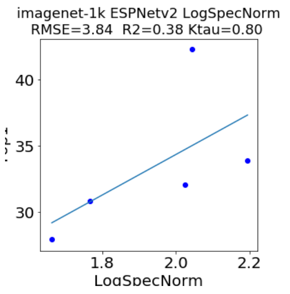


(l) imagenet-1k–SqueezeNext, $\langle \log \| \cdot \|_\alpha^\alpha \rangle$

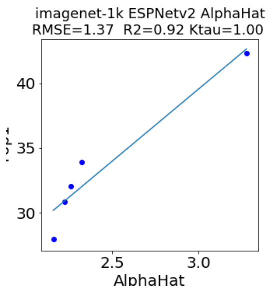
Figure 11: Regression plots for model-dataset pairs, based on data from Table 7, Table 8, and Table 9. Each row corresponds to a different dataset-model pair: imagenet-1k–HRNet; imagenet-1k–DRN-C; and imagenet-1k–SqueezeNext; repsectively. Each column corresponds to a different metric: $\langle \log \| \cdot \|_F \rangle$; $\langle \log \| \cdot \|_\infty \rangle$; $\hat{\alpha}$; and $\langle \log \| \cdot \|_\alpha^\alpha \rangle$; repsectively.



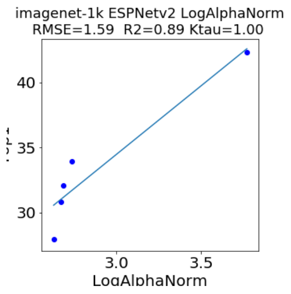
(a) imagenet-1k–ESPNetv2, $\langle \log \|\cdot\|_F \rangle$



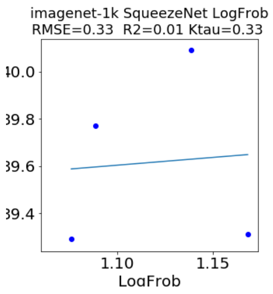
(b) imagenet-1k–ESPNetv2, $\langle \log \|\cdot\|_\infty \rangle$



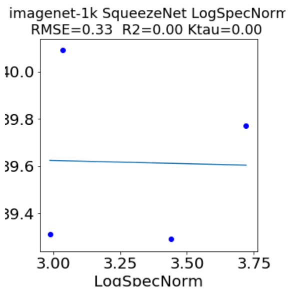
(c) imagenet-1k–ESPNetv2, $\hat{\alpha}$



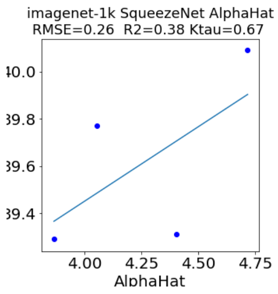
(d) imagenet-1k–ESPNetv2, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



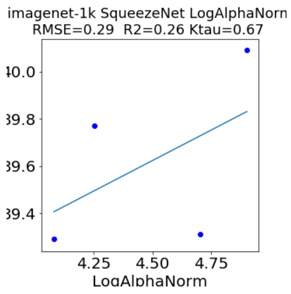
(e) imagenet-1k–SqueezeNet, $\langle \log \|\cdot\|_F \rangle$



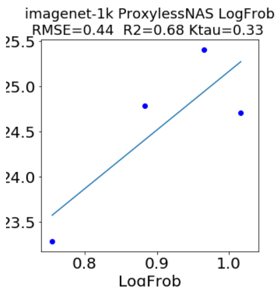
(f) imagenet-1k–SqueezeNet, $\langle \log \|\cdot\|_\infty \rangle$



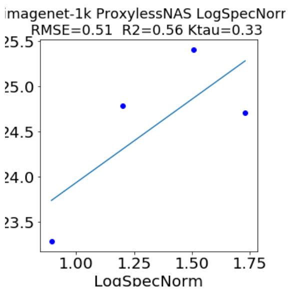
(g) imagenet-1k–SqueezeNet, $\hat{\alpha}$



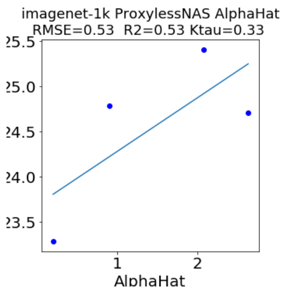
(h) imagenet-1k–SqueezeNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



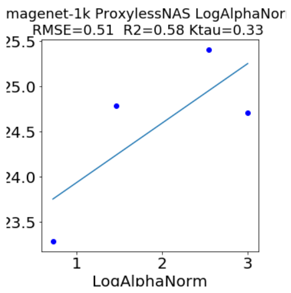
(i) imagenet-1k–ProxylessNAS, $\langle \log \|\cdot\|_F \rangle$



(j) imagenet-1k–ProxylessNAS, $\langle \log \|\cdot\|_\infty \rangle$

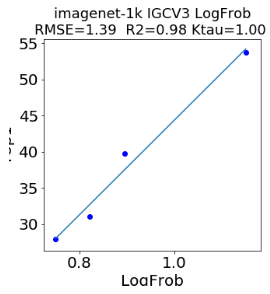


(k) imagenet-1k–ProxylessNAS, $\hat{\alpha}$

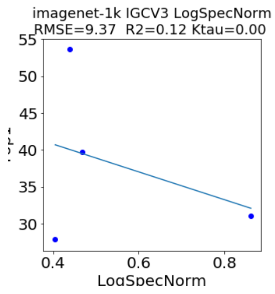


(l) imagenet-1k–ProxylessNAS, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$

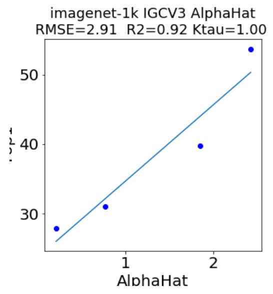
Figure 12: Regression plots for model-dataset pairs, based on data from Table 7, Table 8, and Table 9. Each row corresponds to a different dataset-model pair: imagenet-1k–ESPNetv2; imagenet-1k–SqueezeNet; and imagenet-1k–ProxylessNAS; repsectively. Each column corresponds to a different metric: $\langle \log \|\cdot\|_F \rangle$; $\langle \log \|\cdot\|_\infty \rangle$; $\hat{\alpha}$; and $\langle \log \|\cdot\|_\alpha^\alpha \rangle$; repsectively.



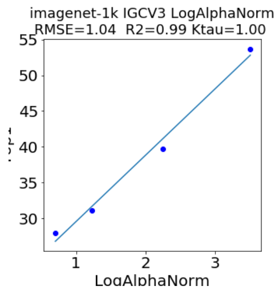
(a) imagenet-1k–IGCV3, $\langle \log \|\cdot\|_F \rangle$



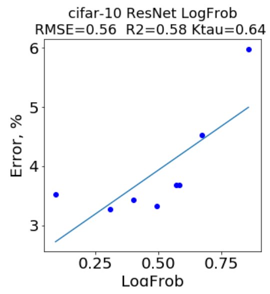
(b) imagenet-1k–IGCV3, $\langle \log \|\cdot\|_\infty \rangle$



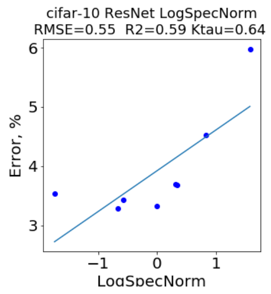
(c) imagenet-1k–IGCV3, $\hat{\alpha}$



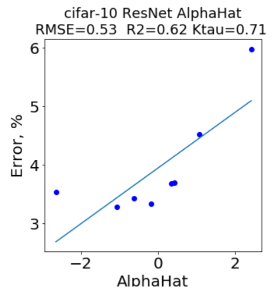
(d) imagenet-1k–IGCV3, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



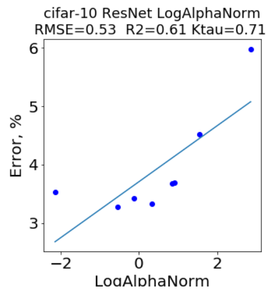
(e) cifar-10–ResNet, $\langle \log \|\cdot\|_F \rangle$



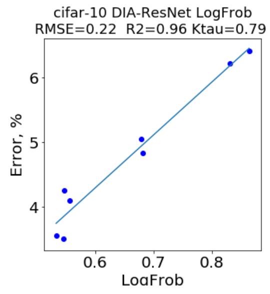
(f) cifar-10–ResNet, $\langle \log \|\cdot\|_\infty \rangle$



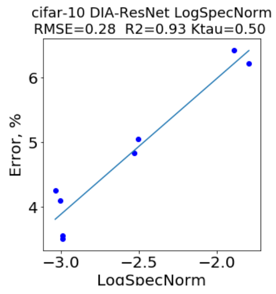
(g) cifar-10–ResNet, $\hat{\alpha}$



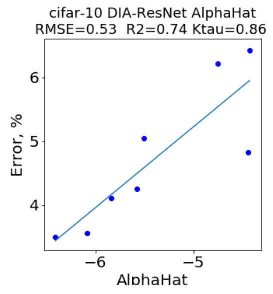
(h) cifar-10–ResNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



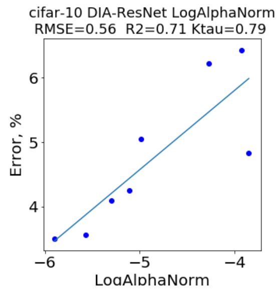
(i) cifar-10–DIA-ResNet, $\langle \log \|\cdot\|_F \rangle$



(j) cifar-10–DIA-ResNet, $\langle \log \|\cdot\|_\infty \rangle$

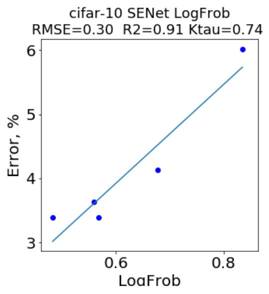


(k) cifar-10–DIA-ResNet, $\hat{\alpha}$

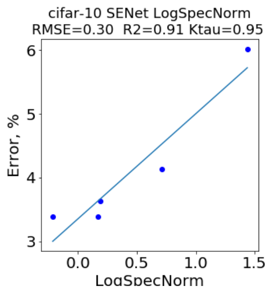


(l) cifar-10–DIA-ResNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$

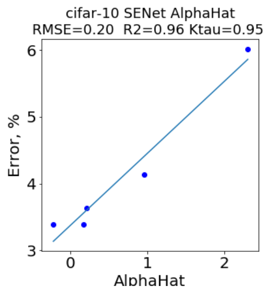
Figure 13: Regression plots for model-dataset pairs, based on data from Table 7, Table 8, and Table 9. Each row corresponds to a different dataset-model pair: imagenet-1k–IGCV3; cifar-10–ResNet; and cifar-10–DIA-ResNet; repsectively. Each column corresponds to a different metric: $\langle \log \|\cdot\|_F \rangle$; $\langle \log \|\cdot\|_\infty \rangle$; $\hat{\alpha}$; and $\langle \log \|\cdot\|_\alpha^\alpha \rangle$; repsectively.



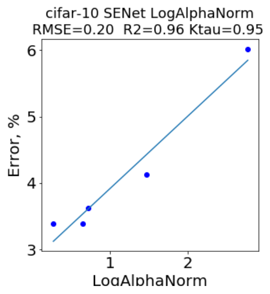
(a) cifar-10–SENet, $\langle \log \| \cdot \|_F \rangle$



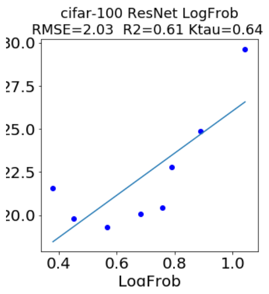
(b) cifar-10–SENet, $\langle \log \| \cdot \|_\infty \rangle$



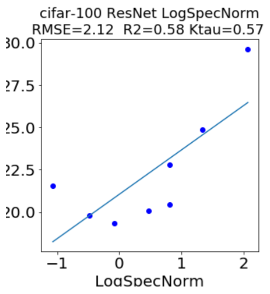
(c) cifar-10–SENet, $\hat{\alpha}$



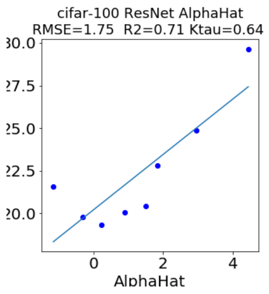
(d) cifar-10–SENet, $\langle \log \| \cdot \|_\alpha^\alpha \rangle$



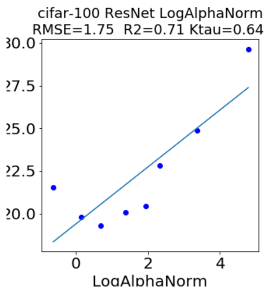
(e) cifar-100–ResNet, $\langle \log \| \cdot \|_F \rangle$



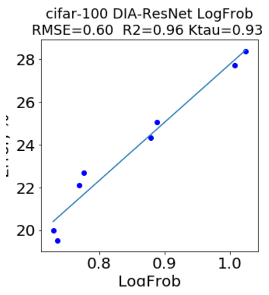
(f) cifar-100–ResNet, $\langle \log \| \cdot \|_\infty \rangle$



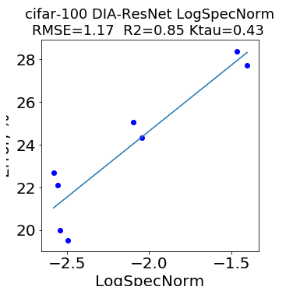
(g) cifar-100–ResNet, $\hat{\alpha}$



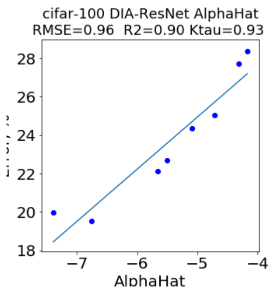
(h) cifar-100–ResNet, $\langle \log \| \cdot \|_\alpha^\alpha \rangle$



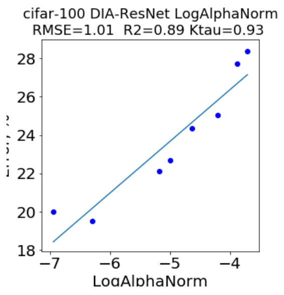
(i) cifar-100–DIA-ResNet, $\langle \log \| \cdot \|_F \rangle$



(j) cifar-100–DIA-ResNet, $\langle \log \| \cdot \|_\infty \rangle$

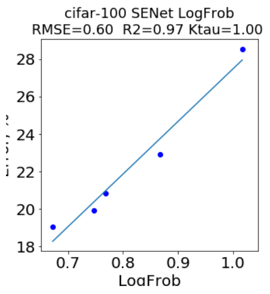


(k) cifar-100–DIA-ResNet, $\hat{\alpha}$

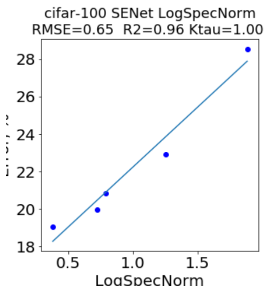


(l) cifar-100–DIA-ResNet, $\langle \log \| \cdot \|_\alpha^\alpha \rangle$

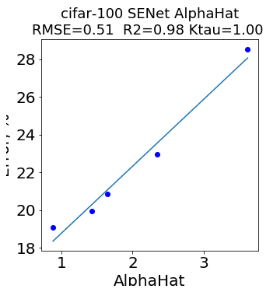
Figure 14: Regression plots for model-dataset pairs, based on data from Table 7, Table 8, and Table 9. Each row corresponds to a different dataset-model pair: cifar-10–SENet; cifar-100–ResNet; and cifar-100–DIA-ResNet; repsectively. Each column corresponds to a different metric: $\langle \log \| \cdot \|_F \rangle$; $\langle \log \| \cdot \|_\infty \rangle$; $\hat{\alpha}$; and $\langle \log \| \cdot \|_\alpha^\alpha \rangle$; repsectively.



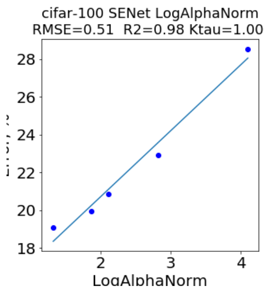
(a) cifar-100–SENet, $\langle \log \|\cdot\|_F \rangle$



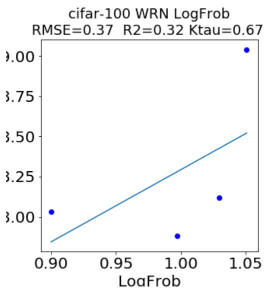
(b) cifar-100–SENet, $\langle \log \|\cdot\|_\infty \rangle$



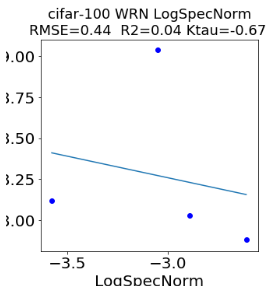
(c) cifar-100–SENet, $\hat{\alpha}$



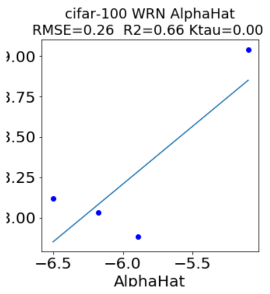
(d) cifar-100–SENet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



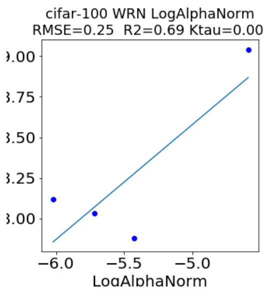
(e) cifar-100–WRN, $\langle \log \|\cdot\|_F \rangle$



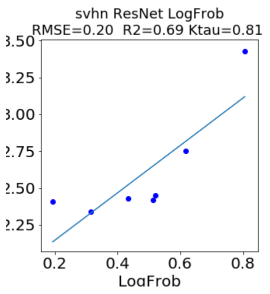
(f) cifar-100–WRN, $\langle \log \|\cdot\|_\infty \rangle$



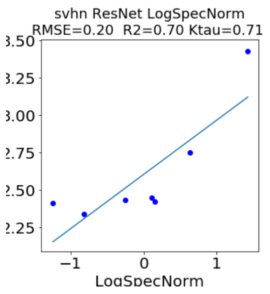
(g) cifar-100–WRN, $\hat{\alpha}$



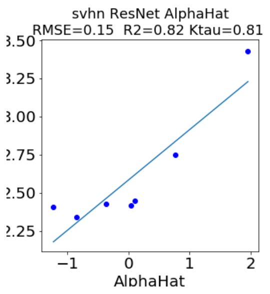
(h) cifar-100–WRN, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



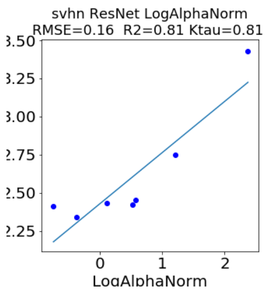
(i) svhn–ResNet, $\langle \log \|\cdot\|_F \rangle$



(j) svhn–ResNet, $\langle \log \|\cdot\|_\infty \rangle$

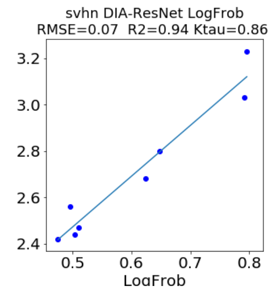


(k) svhn–ResNet, $\hat{\alpha}$

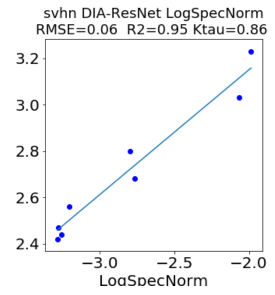


(l) svhn–ResNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$

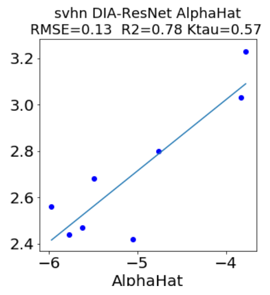
Figure 15: Regression plots for model-dataset pairs, based on data from Table 7, Table 8, and Table 9. Each row corresponds to a different dataset-model pair: cifar-100–SENet; cifar-100–WRN; and svhn–ResNet; repsectively. Each column corresponds to a different metric: $\langle \log \|\cdot\|_F \rangle$; $\langle \log \|\cdot\|_\infty \rangle$; $\hat{\alpha}$; and $\langle \log \|\cdot\|_\alpha^\alpha \rangle$; repsectively.



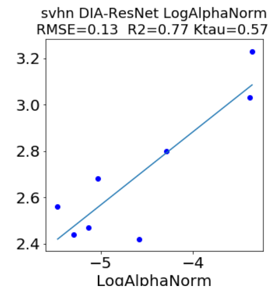
(a) svhn–DIA-ResNet, $\langle \log \|\cdot\|_F \rangle$



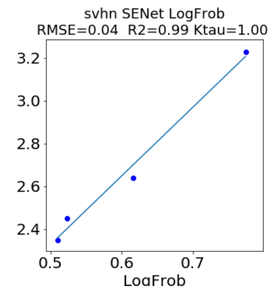
(b) svhn–DIA-ResNet, $\langle \log \|\cdot\|_\infty \rangle$



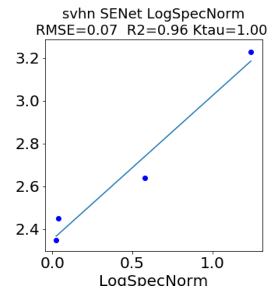
(c) svhn–DIA-ResNet, $\hat{\alpha}$



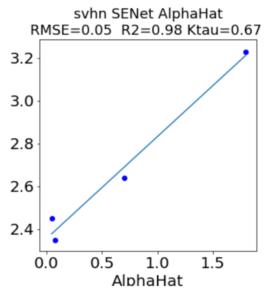
(d) svhn–DIA-ResNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



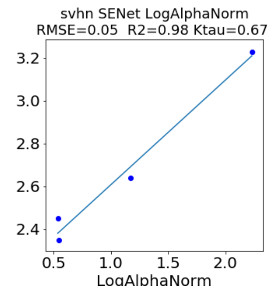
(e) svhn–SENet, $\langle \log \|\cdot\|_F \rangle$



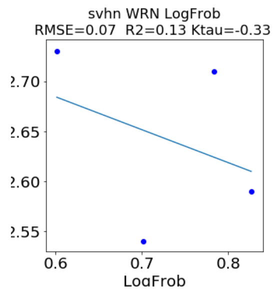
(f) svhn–SENet, $\langle \log \|\cdot\|_\infty \rangle$



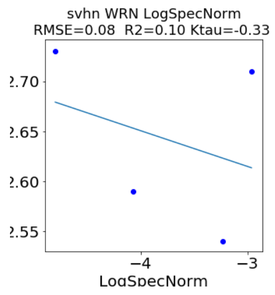
(g) svhn–SENet, $\hat{\alpha}$



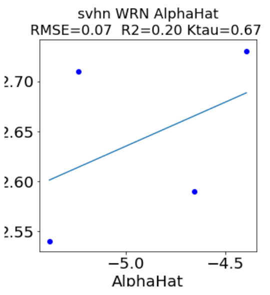
(h) svhn–SENet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



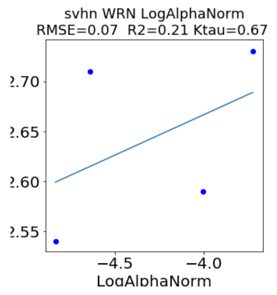
(i) svhn–WRN, $\langle \log \|\cdot\|_F \rangle$



(j) svhn–WRN, $\langle \log \|\cdot\|_\infty \rangle$

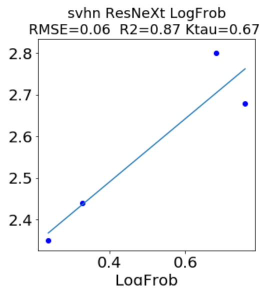


(k) svhn–WRN, $\hat{\alpha}$

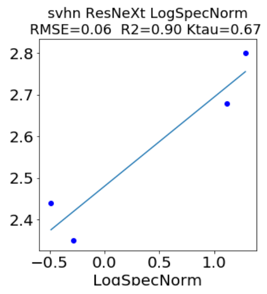


(l) svhn–WRN, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$

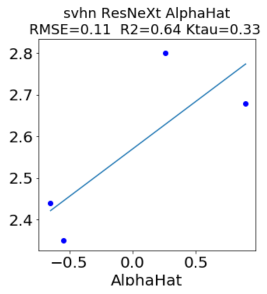
Figure 16: Regression plots for model-dataset pairs, based on data from Table 7, Table 8, and Table 9. Each row corresponds to a different dataset-model pair: svhn–DIA-ResNet; svhn–SENet; and svhn–WRN; repsectively. Each column corresponds to a different metric: $\langle \log \|\cdot\|_F \rangle$; $\langle \log \|\cdot\|_\infty \rangle$; $\hat{\alpha}$; and $\langle \log \|\cdot\|_\alpha^\alpha \rangle$; repsectively.



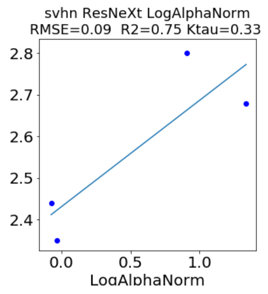
(a) svhn–ResNeXt, $\langle \log \|\cdot\|_F \rangle$



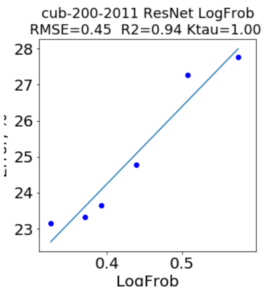
(b) svhn–ResNeXt, $\langle \log \|\cdot\|_\infty \rangle$



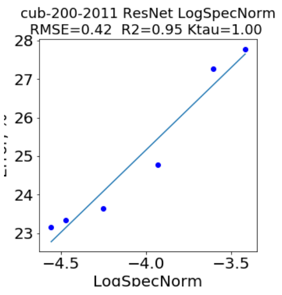
(c) svhn–ResNeXt, $\hat{\alpha}$



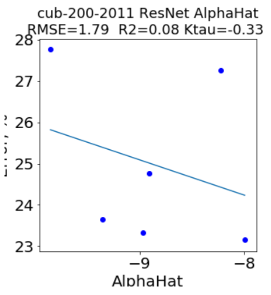
(d) svhn–ResNeXt, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



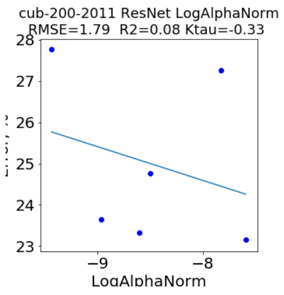
(e) cub-200-2011–ResNet, $\langle \log \|\cdot\|_F \rangle$



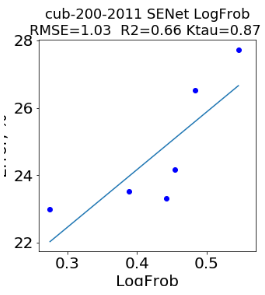
(f) cub-200-2011–ResNet, $\langle \log \|\cdot\|_\infty \rangle$



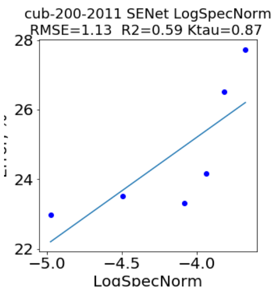
(g) cub-200-2011–ResNet, $\hat{\alpha}$



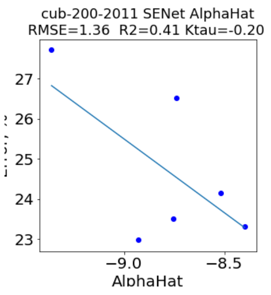
(h) cub-200-2011–ResNet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$



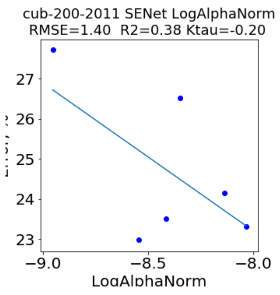
(i) cub-200-2011–SENet, $\langle \log \|\cdot\|_F \rangle$



(j) cub-200-2011–SENet, $\langle \log \|\cdot\|_\infty \rangle$



(k) cub-200-2011–SENet, $\hat{\alpha}$



(l) cub-200-2011–SENet, $\langle \log \|\cdot\|_\alpha^\alpha \rangle$

Figure 17: Regression plots for model-dataset pairs, based on data from Table 7, Table 8, and Table 9. Each row corresponds to a different dataset-model pair: svhn–ResNeXt; cub-200-2011–ResNet; and cub-200-2011–SENet; repsectively. Each column corresponds to a different metric: $\langle \log \|\cdot\|_F \rangle$; $\langle \log \|\cdot\|_\infty \rangle$; $\hat{\alpha}$; and $\langle \log \|\cdot\|_\alpha^\alpha \rangle$; repsectively.