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 practical applications, one works with deep neural network (DNN) models trained by someone else. For such *pretrained models*, one typically does not have access to training data or test data. Moreover, one does not know many details about the model, such as the specifics of the training data, the loss function, the hyperparameter values, etc. Given one or many pretrained models, can one say anything about the expected performance or quality of the models? Here, we present and evaluate empirical quality metrics for pretrained DNN models at scale. Using the open-source *WeightWatcher* tool, we analyze hundreds of publicly-available pretrained models, including older and current state-of-the-art models in computer vision (CV) and natural language processing (NLP). We examine both familiar norm-based capacity control metrics (Frobenius and Spectral norms) as well as newer Power Law (PL) based metrics (including fitted PL exponents, α, and the Weighted Alpha metric, $\hat{\alpha}$), from the recently-developed Theory of Heavy-Tailed Self Regularization (HT-SR). We also introduce the α-Shatten Norm metric. We find that norm-based metrics correlate well with reported test accuracies for well-trained models across nearly all CV architecture series. On the other hand, we find that norm-based metrics can not distinguish "good-versus-bad" models—which, arguably is the point of needing quality metrics. Indeed, they may give spurious results. We also find that PL-based metrics do much better—quantitatively better at discriminating among a series of "good-better-best" models, and qualitatively better at discriminating "good-versus-bad" models. PL-based metrics can also be used to characterize fine-scale properties of these models, and we introduce the layer-wise *Correlation Flow* as new quality assessment. We show how poorly-trained (and/or poorly fine-tuned) models may exhibit both *Scale Collapse* and unusually large PL exponents, $\alpha \gg 6$, in particular for recent NLP models. Our techniques, as implemented in the *WeightWatcher* tool, can be used to identify when a pretrained DNN has problems that can not 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 less well-known problem, but one that is increasingly important, in particular in industrial-scale artificial

*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.

intelligence (AI), arises when the model *user* is not the model *developer*. Here, one may not have access to either 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 siloed: some groups will gather data, other groups will develop models, and other groups will use those models. Users of models can not 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: do we have enough data for this 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. We seek a practical theory for pretrained models which can predict how, when, and why such models can be expected to perform well or poorly.

In this paper, we present and evaluate quality metrics for pretrained deep neural network (DNN) models, and we do so at scale. 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., there are now hundreds of pretrained models in CV (≥ 500) and NLP (≈ 100).[1] These provide a large corpus of models that by some community standard are state-of-the-art.[2] Importantly, all of these models have been trained by someone else and have been viewed to be of sufficient interest/quality to be made publicly-available; and, for all of these models, we have no access to training data or testing data, and we have no knowledge of the training/testing protocols.

The *quality metrics* we consider are based on the spectral properties of the layer weight matrices. They are based on norms of weight matrices (such norms have been used in traditional statistical learning theory to bound capacity and construct regularizers) and/or parameters of power law (PL) fits of the eigenvalues of weight matrices (such PL fits are based on statistical mechanics approaches to DNNs). Note that, 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 error during training),

[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.

[2] Clearly, there is a selection bias or survivorship bias here—people tend not to make publicly-available their poorly-performing models—but these models are things in the world that (like social networks or the internet) can be analyzed for their properties.

we do not seek a new regularizer, and *we do not aim to evaluate a single model* (e.g., as with hyperparameter optimization).[3] Instead, we want to examine different models across common architecture series, and we want to compare models between different architectures themselves, and in both cases, we ask:

> *Can we predict trends in the quality of pretrained DNN models without access to training or testing data?*

To answer this question, we analyze hundreds of publicly-available pretrained state-of-the-art CV and NLP models. Here is a summary of our main results.

- **Norm-based metrics and well-trained models.** Norm-based metrics do a reasonably good job at predicting quality trends in well-trained CV/NLP models.
- **Norm-based metrics and poorly-trained models.** Norm-based metrics may give spurious results when applied to poorly-trained models (e.g., models trained without enough data, etc.), exhibiting *Scale Collapse* for these models.
- **PL-based metrics and model quality.** PL-based metrics do much better at predicting quality trends in pretrained CV/NLP models. They are quantitatively better at discriminating "good-better-best" trends, and qualitatively better at distinguishing "good-versus-bad" models.
- **PL-based metrics and model diagnostics.** PL-based metrics can also be used to characterize fine-scale model properties (including 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.).

We emphasize that our goal is a practical theory to predict trends in the quality of state-of-the-art DNN models, i.e., not to make a statement about every publicly-available model. We have examined hundreds of models, and we identify general trends, but we also highlight interesting exceptions.

The WeightWatcher Tool. All of our computations were performed with the publicly-available *WeightWatcher* tool (version 0.2.7) [1]. To be fully reproducible, we only examine publicly-available, pretrained models, and we also provide all Jupyter and Google Colab notebooks used in an accompanying github repository [2]. See Appendix A for details on how to reproduce all results.

Organization of this paper. We start in Section 2 and Section 3 with background and an overview of our general approach. In Section 4, we study three well-known widely-available DNN CV architectures (the VGG, ResNet, and DenseNet series of models); and we provide an illustration of our basic methodology, both to evaluate the different metrics against reported test accuracies and to use quality metrics to understand model properties. Then, in Section 5, we look at several variations of a popular NLP DNN architecture (the OpenAI GPT and GPT2 models); and we show how model quality and properties vary between several variants of GPT and GPT2, including how metrics behave similarly and differently. Then, in Section 6, we present results based on an analysis of hundreds of pretrained DNN models, showing how well each metric predicts the reported test accuracies, and how the PL-based metrics perform remarkably well. Finally, in Section 7, we provide a brief discussion and conclusion.

[3] One could of course use these techniques to improve training, and we have been asked about that, but we are not interested in that here. Our main goal here is to use these techniques to evaluate properties of state-of-the-art pretrained DNN models.

2 Background and Related Work

Most theory for DNNs is applied to small toy models and assumes access to data. There is very little work asking how to predict, in a theoretically-principled manner, the quality of large-scale state-of-the-art DNNs, and how to do so without access to training data or testing data or details of the training protocol, etc. Our approach is, however, related to two other lines of work.

Statistical mechanics theory for DNNs. Statistical mechanics ideas have long had influence on DNN theory and practice [3, 4, 5]; and our best-performing metrics (those using fitted PL exponents) are based on statistical mechanics [4, 6, 7, 8, 9], in particular the recently-developed *Theory of Heavy Tailed Self Regularization (HT-SR)* [6, 7, 9]. We emphasize that the way in which we (and HT-SR Theory) *use* statistical mechanics theory is quite different than the way it is more commonly formulated. Several very good overviews of the more common approach are available [3, 5]. We *use* statistical mechanics in a broader sense, drawing upon techniques from quantitative finance and random matrix theory. Thus, much more relevant for our methodological approach is older work of Bouchaud, Potters, Sornette, and coworkers [10, 11, 12, 13] on the statistical mechanics of heavy tailed and strongly correlated systems.

Norm-based capacity control theory. There is also a large body of work on using norm-based metrics to bound generalization error [14, 15, 16]. In this area, theoretical work aims to prove generalization bounds, and applied work uses these norms to construct regularizers to improve training. While we do find that norms provide relatively good quality metrics, at least for distinguishing good-better-best among well-trained models, we are not interested in proving generalization bounds or developing new regularizers.

3 Methods

Let us write the Energy Landscape (or optimization function, parameterized by $\mathbf{W}_l$s and $\mathbf{b}_l$s) for a DNN with L layers, activation functions $h_l(\cdot)$, and $N \times M$ weight matrices $\mathbf{W}_l$ and biases $\mathbf{b}_l$, as:

$$E_{DNN} = h_L(\mathbf{W}_L \times h_{L-1}(\mathbf{W}_{L-1} \times h_{L-2}(\cdots) + \mathbf{b}_{L-1}) + \mathbf{b}_L). \tag{1}$$

Each DNN layer contains one or more layer 2D $N \times M$ weight matrices, $\mathbf{W}_l$, or pre-activation maps, $\mathbf{W}_{i,l}$, extracted from 2D Convolutional layers, and where $N > M$.[4] (We may drop the i and/or i, l subscripts below.) See Appendix A for how we define the Conv2D layer matrixes and for our choices of normalization.

Assume we are given several pretrained DNNs, e.g., as part of an architecture series. The models have been trained and evaluated on labeled data $\{d_i, y_i\} \in \mathcal{D}$, using standard techniques. The pretrained pytroch model files are publicly-available, and the test accuracies have been reported online. In this study, we do not have access to this data, and we have not trained any of the models ourselves, nor have we re-evaluated the test accuracies. 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 $\mathbf{W}_l$.

[4] We do not use intra-layer information from the models in our quality metrics, but (as we will describe) our metrics can be used to learn about intra-layer model properties.

DNN Empirical Quality Metrics. The best performing empirical quality metrics depend on the norms and/or spectral properties of each weight matrix, $\mathbf{W}$, and/or, equivalently, it's *Empirical Correlation Matrix*: $\mathbf{X} = \mathbf{W}^T\mathbf{W}$.

Here, we consider the following metrics.

- Frobenius Norm: $\|\mathbf{W}\|_F^2 = \|\mathbf{X}\|_F = \sum_{i=1}^{M} \lambda_i$
- Spectral Norm: $\|\mathbf{W}\|_\infty^2 = \|\mathbf{X}\|_\infty = \lambda_{max}$
- Weighted Alpha: $\hat{\alpha} = \alpha \log \lambda_{max}$
- α-Norm (or α-Shatten Norm):[5] $\|\mathbf{X}\|_\alpha^\alpha = \sum_{i=1}^{M} \lambda_i^\alpha$

Here, λ_i is the i^{th} eigenvalue of the $\mathbf{X}$, and λ_{max} is the maximum eigenvalue. Recall that the eigenvalues are squares of the singular values σ_i of $\mathbf{W}$: $\lambda_i = \sigma_i^2$. Also, note that we do *not* normalize $\mathbf{X}$ by $1/N$; see Appendix A for a discussion of this issue.

The first two norms are well-known in ML; the last two deserve special mention. The empirical parameter α is the Power Law (PL) exponent that arises in the recently-developed HT-SR Theory [6, 7, 9]. Operationally, α is determined by using the publicly-available *WeightWatcher* tool [1] to fit the Empirical Spectral Density (ESD) of $\mathbf{X}$, i.e., a histogram of the eigenvalues, call it $\rho(\lambda)$, to a truncated PL,

$$\rho(\lambda) \sim \lambda^\alpha, \quad \lambda \leq \lambda_{max}. \tag{2}$$

Each of these quantities is defined for a given layer $\mathbf{W}$ matrix.

For norm-based metrics, we use the average of the log norm, and 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),

$$\mathcal{C} \sim \|\mathbf{W}_1\| \times \|\mathbf{W}_2\| \times \cdots \times \|\mathbf{W}_L\|, \tag{3}$$

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

$$\log \mathcal{C} \sim \log \|\mathbf{W}_1\| + \log \|\mathbf{W}_2\| + \cdots + \log \|\mathbf{W}_L\|, \tag{4}$$

takes the form of an average Log Norm. For the *Frobenius Norm metric* and *Spectral Norm metric*, we can use Eqn. (4) directly.[6]

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,

$$\hat{\alpha} = \frac{1}{L} \sum_l \alpha_l \log \lambda_{max,l} \approx \langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle, \tag{5}$$

where L is the total number of layer weight matrices. The Weighted Alpha metric was introduced previously [9], where it was shown to correlate well with trends in reported test accuracies of pretrained DNNs, albeit on a limited set of models.

Based on this, in this paper, we introduce and evaluate the *α-Shatten Norm metric*. Notice for the α-Shatten Norm metric, however, α_l varies from layer to layer, and so in Eqn. (6) it can not be taken out of the sum:

[5] Notice $\|\mathbf{W}\|_{2\alpha}^{2\alpha} = \|\mathbf{X}\|_\alpha^\alpha$. We use $\mathbf{X}$ to emphasize that α depends on the ESD of $\mathbf{X}$.

[6] 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.

$$\sum_l \log \|\mathbf{X}_l\|_{\alpha_l}^{\alpha_l} = \sum_l \alpha_l \log \|\mathbf{X}_l\|_{\alpha_l}. \tag{6}$$

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 [17]; and the Weighted Alpha and α-Shatten norm metrics often behave like an improved, weighted average Log Spectral Norm, and may track this metric in some cases.

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}. (Technically, we also need the minimum eigenvalue λ_{min}, but this detail does not affect our analysis.) The PL exponent α measures of 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 for the entire DNN.

Convolutional Layers and Normalization issues. There are several technical issues (regarding spectral analysis of convolutional layers and normalization of empirical matrices) that are important for reproducibility of our results. See Appendix A for a discussion.

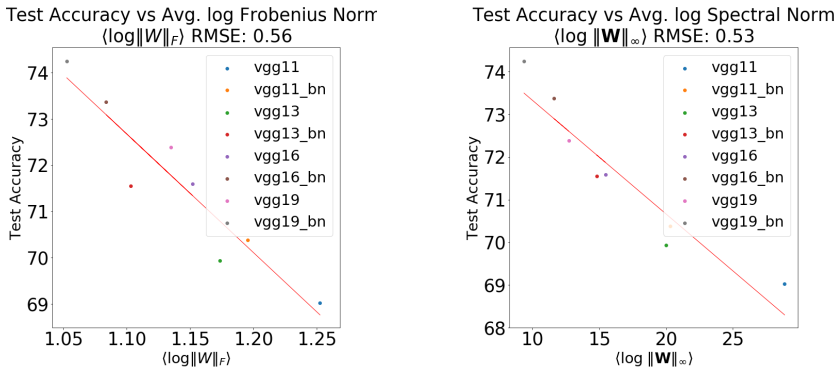
4 Comparison of CV models

In this section, we examine empirical quality metrics described in Section 3 for several CV model architecture series. This includes the VGG, ResNet, and DenseNet series of models, each of which consists of several pretrained DNN models, trained on the full ImageNet [18] dataset, and each of which is distributed with the current open source pyTorch framework (version 1.4) [19]. This also includes a larger set of ResNet models, trained on the ImageNet-1K dataset [18], provided on the OSMR "Sandbox for training convolutional networks for computer vision" [20], which we call the ResNet-1K series.

We perform *coarse model analysis*, comparing and contrasting the four model series, and predicting trends in model quality. We also perform *fine layer analysis*, as a function of depth for these models, illustrating that PL-based metrics can provide novel insights among the VGG, ResNet/ResNet-1K, and DenseNet architectures.

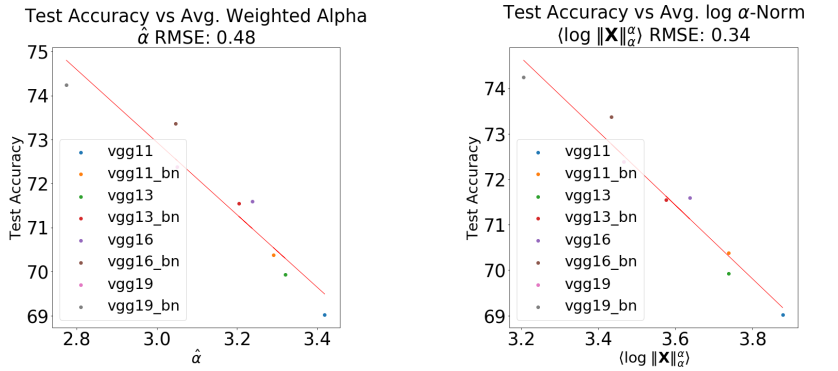
Average Quality Metrics versus Reported Test Accuracies. We have examined the performance of the four quality metrics (Log Frobenius norm, Log Spectral norm, Weighted Alpha, and Log α-Norm) applied to each of the VGG, ResNet, ResNet-1K, and DenseNet series. To start, Figure 1 considers the VGG series (in particular, the pretrained models VGG11, VGG13, VGG16, and VGG19, with and without BN), and it plots the four quality metrics versus the reported test accuracies [19],[7] as well as a basic linear regression line. All four metrics correlate quite well with the reported Top1 accuracies, with smaller norms and smaller values of $\hat{\alpha}$ implying better generalization (i.e., greater accuracy, lower error). While all four metrics perform well, notice that the Log α-Norm metric ($\log \|\mathbf{W}\|_{\alpha}^{\alpha}$) performs best (with an RMSE of 0.42, see Table 1); and the Weighted Alpha metric ($\hat{\alpha} = \alpha \log \lambda_{max}$), which is an approximation to the Log α-Norm metric [17], performs second best (with an RMSE of 0.48, see Table 1).

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



(a) Log Frobenius Norm, VGG

(b) Log Spectral Norm, VGG



(c) Weighted Alpha, VGG

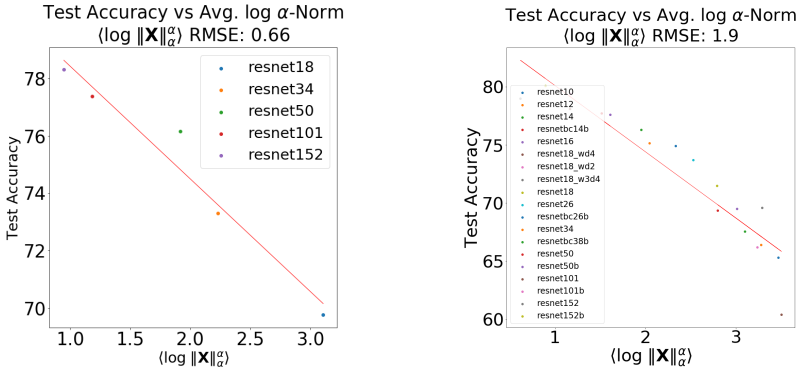
(d) Log α-Norm, VGG

Figure 1: Comparison of Average Log Norm and Weighted Alpha quality metrics versus reported test accuracy for pretrained VGG models (with and without BN), trained on ImageNet, available in pyTorch (v1.4). Metrics fit by linear regression, RMSE reported.

See Table 1 for a summary of results for Top1 accuracies for all four metrics for the VGG, ResNet, and DenseNet series. Similar results (not shown) are obtained for the Top5 accuracies. Overall, for the the ResNet, ResNet-1K, and DenseNet series, all metrics perform relatively well, the Log α-Norm metric performs second best, and the Weighted Alpha metric performs best. These model series are all well-trodden, and our results indicate that norm-based metrics and PL-based metrics can both distinguish among a series of "good-better-best" models, with PL-based metrics performing somewhat (i.e., quantitatively) better.

The DenseNet series has similar behavior to what we see in Figures 1 and 2 for the other models. However, as noted in Table 1, it has only 4 data points. In our larger analysis, in Section 6, we will only include series with 5 or more models. (Note that these and many other such plots can be seen on our publicly-available repo.)

Variation in Data Set Size. We are interested in how our four quality metrics depend on data set size. To examine this, we look at results on ResNet versus ResNet-1K. See Figure 2, which plots and compares the Log α-Norm metric for the full ResNet model, trained on the full ImageNet dataset, against the ResNet-1K model, which has been trained on a much smaller ImageNet-1K data set. The Log α-Norm is much better than the Log Frobenius/Spectral norm metrics (although, as Table 1 shows, it is actually slightly worse than the Weighted Alpha metric). The ResNet series has strong correlation, with an RMSE of 0.66, whereas the ResNet-1K series



(a) ResNet, Log α-Norm

(b) ResNet-1K, Log α-Norm

Figure 2: Comparison of Average α-Norm quality metric ($\langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle$) versus reported Top1 test accuracy for the ResNet and ResNet-1K pretrained (pyTorch) models.

Series	#	$\langle \log \|\mathbf{W}\|_F \rangle$	$\langle \log \|\mathbf{W}\|_\infty \rangle$	$\hat{\alpha}$	$\langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle$
VGG	6	0.56	0.53	0.48	**0.42**
ResNet	5	0.9	1.4	**0.61**	0.66
ResNet-1K	19	2.4	3.6	**1.8**	1.9
DenseNet	4	0.3	0.26	**0.16**	0.21

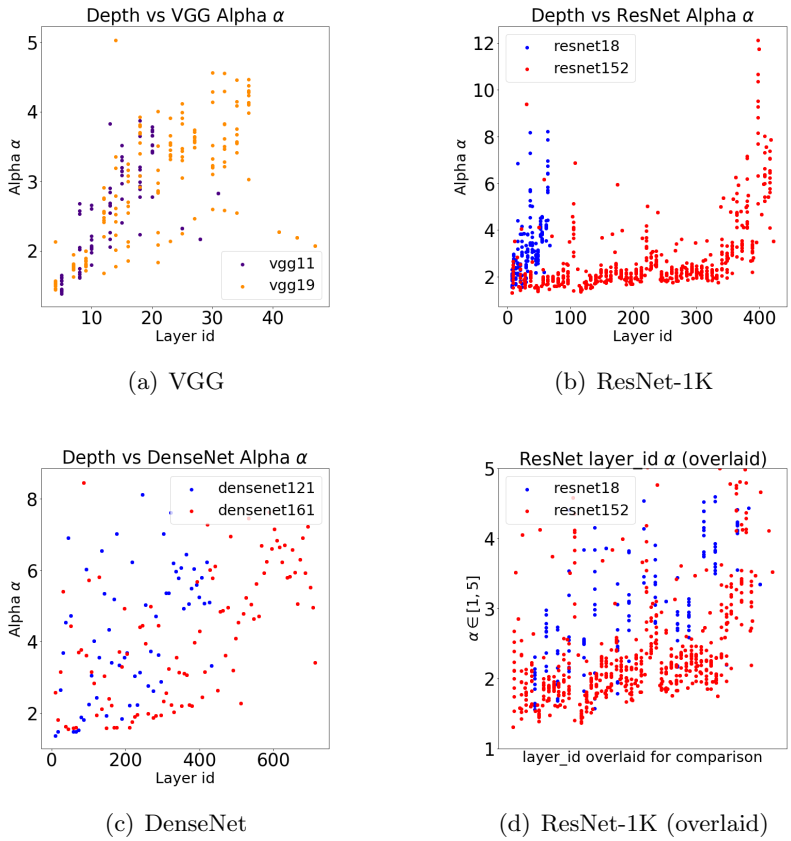
Table 1: RMSE (smaller is better) for linear fits of quality metrics to reported Top1 test error for pretrained models in each architecture series. Column # refers to number of models. VGG, ResNet, and DenseNet were pretrained on ImageNet, and ResNet-1K was pretrained on ImageNet-1K.

also shows good correlation, but has a much larger RMSE of 1.9. (Other metrics exhibit similar behavior.) As expected, the higher quality data set shows a better fit, even with fewer data points.

Layer Analysis: Metrics as a Function of Depth. We can learn much more about a pretrained model by going beyond average values of quality metrics to examining quality metrics for each layer weight matrix, $\mathbf{W}$, as a function of depth (or layer id). For example, we can plot (just) the PL exponent, α, for each layer, as a function of depth. See Figure 3, which plots α for each layer (the first layer corresponds to data, the 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. (Again, a much more detailed set of plots is available at our repo; but note that the corresponding layer-wise plots for Frobenius and Spectral norms are much less interesting than the results we present here.)

In the VGG models, Figure 3(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 $\alpha \lesssim 2.0$ 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 in this figure), 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 3(b) shows that α also increases in the last few layers (more dramatically, in fact, than for VGG, observe the differing scales on the Y axes). However,



(a) VGG (b) ResNet-1K

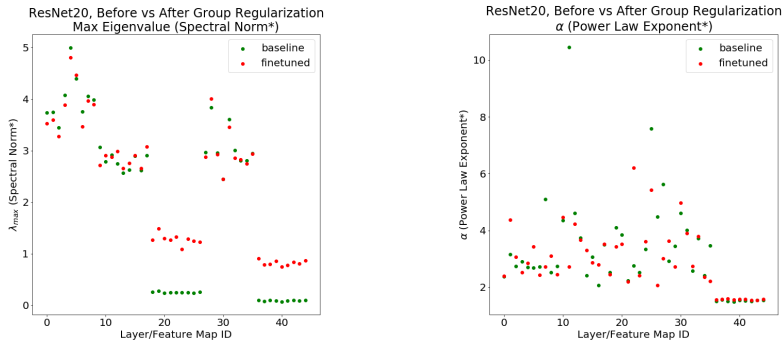
(c) DenseNet (d) ResNet-1K (overlaid)

Figure 3: 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*.

as the ResNet-1K models get deeper, there is a wide range over which α values tend to remain quite 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 3(c) shows that α tends to increase as the layer id increases, in particular for layers toward the end. While this is similar to what is seen in 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.

Comparison of VGG, ResNet, and DenseNet Architectures. We can interpret these observations by recalling the architectural differences between the VGG, ResNet, and DenseNet architectures, and, in particular, the number of of residual connections. VGG resembles the traditional convolutional architectures, such as LeNet5, and consists of several [Conv2D-Maxpool-



(a) λ_{max} for ResNet20 layers

(b) α for ResNet20 layers

Figure 4: 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.

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 GPU memory requirements), and ResNet models of up to 1000 layers have been trained [21].

We conjecture that the efficiency and effectiveness of ResNet is 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 strongly optimized. Contrast this with the DenseNet models, which contains many connections between every layer. Our results (large α, meaning they 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.

Correlation Flow. More generally, we can understand the results presented in Figure 3 in terms of what we will call the *Correlation Flow* of the model. Recall that the average Log α-Norm metric and the Weighted Alpha metric are based on HT-SR Theory [6, 7, 9], which is in turn based on ideas from the statistical mechanics of heavy tailed and strongly correlated systems [10, 11, 12, 13]. There, one expects the weight matrices of well-trained DNNs will exhibit correlations over many size scales. Their 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. Previous work argues this by computing the gradients over the input data. In the absence of training/test data, one might hope that this leaves empirical signatures on weight matrices, and thus we can to try to quantify this by measuring the PL properties of weight matrices. In this case, smaller α values correspond to layers in which correlations across multiple scales are better captured [6, 11], and we expect that small α values that are stable across multiple layers enable better *correlation flow* through the network. We have seen this in many models, including those shown in Figure 3.

Scale Collapse; or How Distillation May Break Models. The similarity between norm-based metrics and PL-based metrics suggests a question: is the Weighted Alpha metric just a variation of the more familiar norm-based metrics? More generally, do fitted α values contain information not captured by norms? In examining hundreds of pretrained models, we have found

several anomalies that demonstrate the power of our approach. In particular, to show that α does capture something different, consider the following example, which looks at a compressed/distilled DNN model [22]. In this example, we show that some distillation methods may actually *break* models unexpectedly by introducing what we call *Scale Collapse*, where several distilled layers have unexpectedly small Spectral Norms.

We consider ResNet20, trained on CIFAR10, before and after applying the Group Regularization distillation technique, as implemented in the `distiller` package [23]. 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 2) suggest that the baseline Spectral Norms should be *smaller* on average than the fine-tuned ones. *The opposite is observed.* Figure 4 presents the Spectral Norm (here denoted $\log \lambda_{max}$) and PL exponent (α) for each individual layer weight matrix $\mathbf{W}$.[8] On the other hand, the α values (in Figure 4(b)) do not differ systematically between the baseline and fine-tuned models. Also (not shown), the average (unweighted) baseline α is *smaller* than the fine-tuned average (as predicted by HT-SR Theory, the basis of $\hat{\alpha}$).

That being said, Figure 4(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.

The 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. While norms make good regularizers for a single model, there is no reason *a priori* to expect them correlate so well with test accuracies across different models. We do expect, however, the PL α to do so because it effectively measures the amount of correlation in the model [6, 7, 9]. The reason for the anomalous behavior shown in Figure 4 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.

5 Comparison of NLP Models

In this section, we examine empirical quality metrics described in Section 3 for several NLP model architectures. Within the past two 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 [24]. In contrast to smaller pre-Activation maps arising in Cond2D layers, Attention matrices are significantly larger. In general, we have found that 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 [10, 11]), this suggests that these models fail to capture successfully many of the correlations in the data (relative to their size) and thus are substantially *under*-trained. More generally, compared to the CV models of Section 4, modern NLP models have larger weight matrices and display different spectral properties. Thus, they provide a very different test for our empirical quality metrics.

While norm-based metrics perform reasonably well on well-trained NLP models, they often behave anomalously on poorly-trained models. Indeed, for such "bad" models, weight matrices

[8] Here, we only include layer matrices or feature maps with $M \geq 50$.

may display rank collapse, decreased Frobenius mass, or unusually small Spectral norms. (This may be misinterpreted as "smaller is better.") In contrast, PL-based metrics, including the Log α-Norm metric ($\log \|\mathbf{W}\|_\alpha^\alpha$) and the Weighted Alpha metric ($\hat{\alpha} = \alpha \log \lambda_{max}$) display consistent behavior, even on poorly trained models. Indeed, we can use these metrics to help identify when architectures need repair and when more and/or better data are needed.

What do large values of α mean? Many NLP models, such as GPT and BERT, have some 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 MLE estimator it uses is unreliable in this range (i.e., the specific α values returned by the truncated PL fits are less reliable, but having large versus small values of α is reliable). Phenomenologically, if we examine the ESD visually, we can usually describe these $\mathbf{W}$ as in the *Bulk-Decay* or *Bulk-plus-Spikes* phase [6, 7]. Previous work [6, 7] has conjectured that very well-trained DNNs would not have many *outlier* $\alpha \gg 6$; and improved versions of GPT (shown below) and BERT (not shown) confirm this.

OpenAI GPT Models. The OpenAI GPT and GPT2 models provide us with the opportunity to analyze two effects: training the same model with different data set sizes; and increasing the sizes of both the data set and the architectures simultaneously. These models have the remarkable ability to generate fake text that appears to the human to be real, and they have generated significant 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 the same architecture and number of layers as GPT, but which has been trained on a larger and better data set (and with other changes), making it remarkably good at generating (near) human-quality fake text. By comparing the poorly-trained (i.e., "bad") GPT to the well-trained (i.e., "good") GPT2-small, we can identify empirical indicators for when a model has in fact been poorly-trained and thus may perform poorly when deployed. By comparing GPT2-medium to GPT2-large to GPT2-xl, we can examine the effect of increasing data set and model size simultaneously, an example of what we call a series of "good-better-best" models.

The GPT models we analyze are deployed with the popular HuggingFace PyTorch library [25]. 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. The 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.

Average Quality Metrics for GPT and GPT2. We have analyzed the four quality metrics described in Section 3 for the OpenAI GPT and GPT2 pretrained models. See Table 2 for a summary of results. We start by examining trends between GPT and GPT2-small. Observe that all four metrics increase when going from GPT to GPT2-small, i.e., they are smaller for the higher-quality model (higher quality since GPT was trained to better data), when the number of layers is held fixed. Notice that in the GPT model, being poorly trained, the norm metrics all exhibit *Scale Collapse*, compared to GPT2-small.

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. Note that the averages do not include the first embedding layer(s) because they are not (implicitly) normalized.

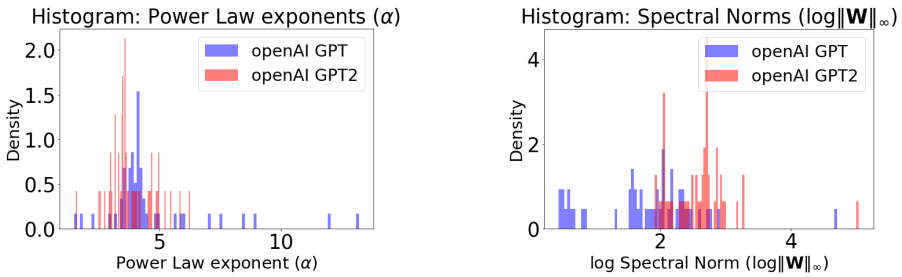
We next examine trends between GPT2-medium to GPT2-large to GPT2-xl. Observe that (with one minor exception involving the log Frobenius norm metric) all four metrics decrease as one goes from medium to large to xl, indicating that the larger models indeed look better than the smaller models. Notice that, for these well-trained models, the norm metrics now behave as expected, decreasing with increasing accuracy.

Going beyond average values, Figure 5(a) shows the histogram (empirical density), for all layers, of α for GPT and GPT2-small. These two histograms are very different. The older deficient GPT has numerous unusually large α exponents—meaning they are not really 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, as expected, 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 α provides a good quality metric for comparing these two models, the "bad" GPT versus the "good" GPT2-small. This should be contrasted with the behavior displayed by the Frobenius norm (not shown) and the Spectral norm.

Scale Collapse in Poorly Trained Models. We next describe the behavior of the Spectral norm in GPT versus GPT2-small. In Figure 5(b), the "bad" GPT model has a smaller mean/median Spectral norm as well as, spuriously, many much smaller Spectral norms, compared to the "good" GPT2-small, violating the conventional wisdom that smaller Spectral norms are better. Indeed, because there are so many anonymously small Spectral norms, it appears that the GPT model may be exhibiting a kind of *Scale Collapse*, like that observed in the distilled CV models (in Figure 4). This is important because it demonstrates that, while the Spectral (or Frobenius) norm may correlate well with predicted test error, it is *not* a good indicator of the overall model quality. It can mispredict good-versus-bad questions in ways not seen with PL-based metrics. Using it as an empirical quality metric may give spurious results when applied to poorly-trained or otherwise deficient models.

(Note that Figure 5(b) also shows some unusually large Spectral Norms. Upon examination, e.g., from Figure 6(b) (below), we see that these correspond to the first embedding layer(s). These layers have a different effective normalization, and therefore a different scale. We discuss this further in Appendix A. Here, we do not include them in our computed average metrics in Table 2, and we do not include them in the histogram plot in Figure 5(b).)

Layer Analysis: Correlation Flow and Scale Collapse in GPT and GPT2. We also examine in Figure 6 the PL exponent α and Log Spectral Norm versus layer id, for GPT and GPT2-small. Let's start with Figure 6(a), which plots α versus the depth (i.e., layer id) for each model. The deficient GPT model displays two trends in α, one stable with $\alpha \sim 4$, and one

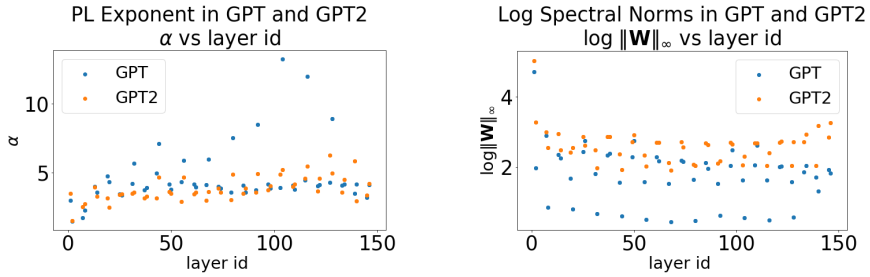


(a) PL exponent (α) (b) Log Spectral Norm ($\log \|\mathbf{W}\|_\infty$)

Figure 5: Histogram of PL exponents (α) and Log Spectral Norms ($\log \|\mathbf{W}\|_\infty$) for weight matrices from the OpenAI GPT and GPT2-small pretrained models.

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 $\alpha \leq 6$. The scale-invariant α metric lets us identify potentially poorly-trained models. These results show that the Correlation Flow differs significantly between GPT and GPT2-small (with the better GPT2-small looking more like the better ResNet-1K from Figure 3(b)).

These results should be contrasted with the corresponding results for Spectral Norms, shown in Figure 6(b). Attention models have two types of layers, one small and large; and the Spectral Norm, in particular, 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 4(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 good-versus-bad models.



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

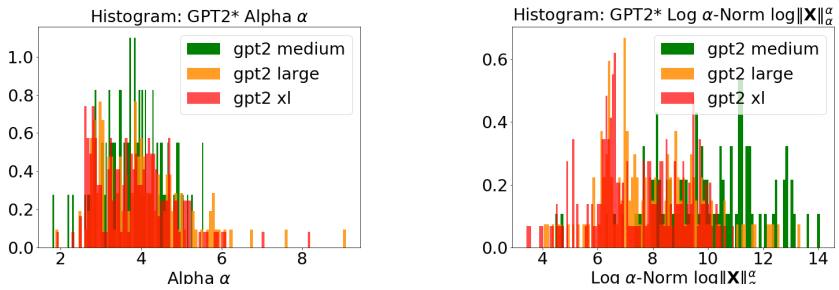
Figure 6: PL exponents (α) (in (a)) and Log Spectral Norms ($\log \|\mathbf{W}\|_\infty$) (in (b)) for weight matrices from the OpenAI GPT and GPT2-small pretrained models. (Note that the quantities being shown on each Y axis are different.) In the text, this is interpreted in terms of *Correlation Flow* and *Scale Collapse*.

GPT2: medium, large, xl. We now look across series of increasingly improving GPT2 models (i.e., we consider good-better-best questions), by examining both the PL exponent α as well as the Log Norm metrics. In general, 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. For

example, see Figure 7, which shows the histograms over the layer weight matrices for fitted PL exponent (α) and the Log Alpha Norm ($\log \|\mathbf{W}\|_\alpha^\alpha$) metric.

We see that the average α decreases with increasing model size, although the differences are less noticeable between the differing good-better-best GTP2 models than between the good-versus-bad GPT and GPT2-small models. Unlike GPT, however, the layer Log Alpha Norms behave more as expected for GPT2 layers, with the larger models consistently having smaller norms. Similarly, the Log Spectral Norm also decreases on average with the larger models (not shown). As expected, the norm metrics can indeed distinguish among good-better-best models among a series well-trained models.

We do notice, however, that while the peaks of the α are getting smaller, towards 2.0, the tails of the distribution shifts right, with larger GPT2 models having more usually large α (also not shown). We suspect this indicates 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 [26].



(a) PL exponent (α)

(b) Log Alpha Norm

Figure 7: Histogram of PL exponents (α) and Log Alpha Norm ($\log \|\mathbf{X}\|_\alpha^\alpha$) 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.)

6 Comparing Hundreds of CV Models

In this section, we summarize results from a large-scale analysis of hundreds of CV models, including models developed for image classification, segmentation, and a range of related tasks. Our aim is to complement the detailed results from Sections 4 and 5 by providing broader conclusions. The models we consider have been pretrained on nine datasets. We provide full details about how to reproduce these results in Appendix A.

We choose ordinary least squares (OLS) regression to quantify the relationship between quality metrics (computed with the *WeightWatcher* tool) and the reported test error and/or accuracy metrics. We 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, as described in Section 4.

Our results are summarized in Table 3. 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

Series	$\log \|\cdot\|_F$	$\log \|\cdot\|_\infty$	$\hat{\alpha}$	$\log \|\cdot\|_\alpha^\alpha$
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**

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 Appendix A).

(and other) results suggest our conclusions from Sections 4 and 5 hold much more generally, and they suggest obvious questions for future work.

7 Conclusion

We have developed (based on strong theory) and evaluated (on a large corpus of publicly-available pretrained models from CV and NLP) methods to predict trends in the quality of state-of-the-art neural networks—without access to training or testing data. Prior to our work, it was not 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. That PL-based metrics perform better (than norm-based metrics) should not be surprising—at least to those familiar with the statistical mechanics of heavy tailed and strongly correlated systems [10, 11, 12, 13] (since our use of PL exponents is designed to capture the idea that well-trained models capture correlations over many size scales in the data). Again, though, our results are the first to demonstrate this. It is also gratifying that our approach can be used to provide fine-scale insight (such as rationalizing the flow of correlations or the collapse of size scale) throughout a network.

We conclude with a few comments on what a *practical theory* of DNNs should look like. To do so, we 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 argues 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. A drawback of the former approach is that it typically makes very strong assumptions on the data, and the strength of those assumptions can limit the practical applicability of the theory. Nearly all of the work on the theory of DNNs 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*.

Acknowledgements. MWM would like to acknowledge ARO, DARPA, NSF, and ONR as well as the UC Berkeley BDD project for providing partial support of this work. We would also like to thank Amir Khosrowshahi and colleagues at Intel for helpful discussion regarding the Group Regularization distillation technique.

References

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

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

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

[4] 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.

[5] 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*, pages 000–000, 2020.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

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

[14] 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.

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

[16] 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.

[17] C. H. Martin and M. W. Mahoney. Unpublished results, 2020.

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

[19] 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.

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

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

[22] 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.

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

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

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

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

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

[28] 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 Appendix

In this appendix, we provide more details on several issues that are important for the reproducibility of our results. All of our computations were performed with the *WeightWatcher* tool (version 0.2.7) [1]. More details and more results are available in an accompanying github repository [2].

A.1 Reproducibility Considerations

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 [27], 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 [28], 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

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

Table 4: Jupyter notebooks used to reproduce all results in Sections 4 and 5.

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).

A.2 Reproducing Sections 4 and 5

We provide a github repository for this paper that includes Jupyter notebooks that fully reproduce all results (as well as many other results) [2]. All results have been produced using the *WeightWatcher* tool (v0.2.7) [1]. The ImageNet and OpenAI GPT pretrained models are provided in the current pyTorch [19] and Huggingface [25] distributions, as specified in the `requirements.txt` file.

A.3 Reproducing Figure 4, for the Distiller Model

In the `distiller` folder of our github repo, we provide the original Jupyter Notebooks, which use the Intel `distiller` framework [23]. Figure 4 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.

A.4 Reproducing Table 3 in Section 6

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 6. 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 [20]. 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 3.

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` repository is constantly being updated with new models. We omit the results for CUB-200-2011,

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

Table 5: Datasets used

Architecture	# of Models
ResNet	30
SENet/SE-ResNet/SE-PreResNet/SE-ResNeXt	24
DIA-ResNet/DIA-PreResNet	18
ResNeXt	12
WRN	12
DLA	6
PreResNet	6
ProxylessNAS	6
VGG/BN-VGG	6
IGCV3	6
EfficientNet	6
SqueezeNext/SqNxt	6
ShuffleNet	6
DRN-C/DRN-D	6
ESPNetv2	6
HRNet z	6
SqueezeNet/SqueezeResNet	6

Table 6: Architectures used

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.

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. Example regressions are shown in Figure 8, as available in the results notebook.

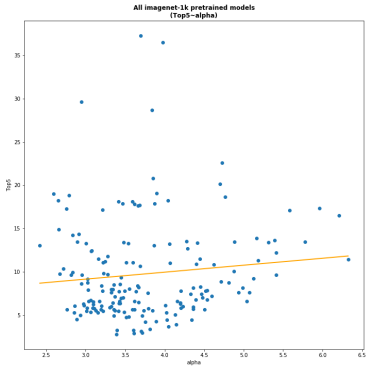
To generate the results in Table 3, we run linear regressions for each architecture series in Table 6, regressing each empirical Log Norm metric against the reported Top1 (and Top5) errors (as listed on the `osmr/imgclsmob` github repository README file [20], with the relevant data extracted and provided 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. See Table 7 and Table 8. 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.

Dataset	Model	$\langle \log \| \cdot \|_F \rangle$	$\langle \log \| \cdot \|_\infty \rangle$	$\hat{\alpha}$	$\langle \log \| \cdot \|_\alpha^\alpha \rangle$
imagenet-1k	ResNet	5.96	11.03	**3.51**	4.01
imagenet-1k	EfficientNet	2.67	**1.23**	2.56	2.50
imagenet-1k	PreResNet	6.59	15.44	**3.59**	3.71
imagenet-1k	ShuffleNet	35.38	89.58	19.54	**18.48**
imagenet-1k	VGG	0.84	**0.68**	1.89	1.59
imagenet-1k	DLA	22.41	**8.49**	14.69	15.68
imagenet-1k	HRNet	0.47	0.51	**0.16**	0.16
imagenet-1k	DRN-C	0.60	0.66	**0.40**	0.48
imagenet-1k	SqueezeNext	21.94	21.39	13.31	**13.23**
imagenet-1k	ESPNetv2	13.77	14.74	**1.87**	2.53
imagenet-1k	IGCV3	1.94	87.76	8.48	**1.09**
imagenet-1k	ProxylessNAS	**0.19**	0.26	0.28	0.26
imagenet-1k	SqueezeNet	0.11	0.11	**0.07**	0.08
cifar-10	ResNet	0.31	0.30	**0.28**	0.28
cifar-10	DIA-ResNet	**0.05**	0.08	0.28	0.32
cifar-10	SENet	0.09	0.09	**0.04**	0.04
cifar-100	ResNet	4.13	4.50	**3.06**	3.06
cifar-100	DIA-ResNet	**0.36**	1.38	0.93	1.02
cifar-100	SENet	0.36	0.43	**0.26**	0.26
cifar-100	WRN	0.14	0.20	0.07	**0.06**
svhn	ResNet	0.04	0.04	**0.02**	0.02
svhn	DIA-ResNet	0.00	**0.00**	0.02	0.02
svhn	SENet	**0.00**	0.00	0.00	0.00
svhn	WRN	0.01	0.01	0.01	**0.01**
svhn	ResNeXt	0.00	**0.00**	0.01	0.01
cub-200-2011	ResNet	0.20	**0.18**	3.19	3.21
cub-200-2011	SENet	**1.07**	1.29	1.85	1.95

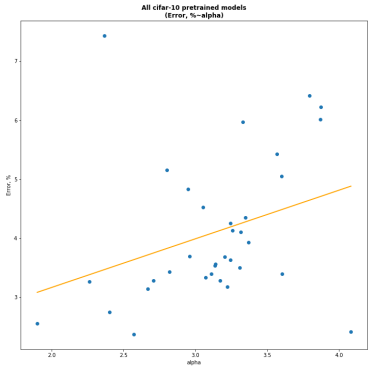
Table 7: MSE Results for all CV model regressions.

Dataset	Model	$\langle \log \| \cdot \|_F \rangle$	$\langle \log \| \cdot \|_\infty \rangle$	$\hat{\alpha}$	$\langle \log \| \cdot \|_\alpha^\alpha \rangle$
imagenet-1k	ResNet	0.82	0.67	**0.90**	0.88
imagenet-1k	EfficientNet	0.65	**0.84**	0.67	0.67
imagenet-1k	PreResNet	0.73	0.36	**0.85**	0.85
imagenet-1k	ShuffleNet	0.63	0.06	0.80	**0.81**
imagenet-1k	VGG	0.71	**0.76**	0.35	0.45
imagenet-1k	DLA	0.13	**0.67**	0.43	0.39
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	IGCV3	0.98	0.12	0.92	**0.99**
imagenet-1k	SqueezeNet	0.01	0.00	**0.38**	0.26
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

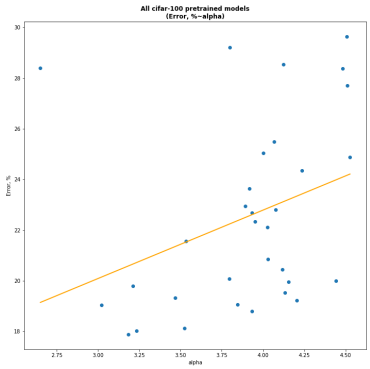
Table 8: R^2 Results for all CV model regressions.



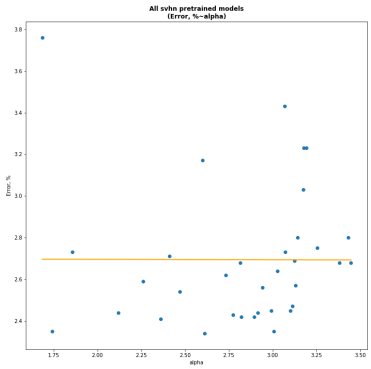
(a) ImageNet 1K



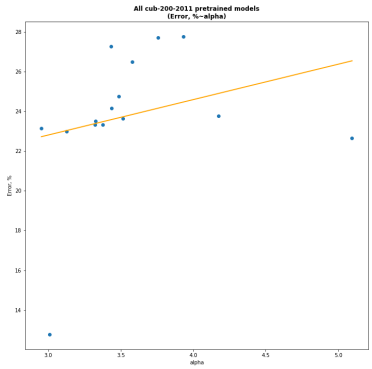
(b) CIFAR 10



(c) CIFAR 100



(d) SVHN



(e) CUB 200

Figure 8: PL exponent α versus reported Top1 Test Accuracies for pretrained DNNs available for five different data sets.