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

Charles H. Martin
Calculation Consulting
San Francisco, CA 94122, USA
charles@CalculationConsulting.com

Serena Peng
Same as me
XXX, XXX
serenapeng7@gmail.com

Michael W. Mahoney
ICSI and Department of Statistics,
University of California at Berkeley
Berkeley, CA 94720, USA
mmahoney@stat.berkeley.edu

ABSTRACT

In many practical applications, one works with deep neural networks (DNNs) models trained by someone else. For such *pretrained models*, one typically does not have access to training data or test data, and, moreover, does not know the details of how the models were built, the specifics of the data that were used to train the model, what was the loss function or hyperparameter values, how precisely the model was regularized, etc. Here, we present and evaluate quality metrics for pretrained neural network models at scale. The most promising are norm-based metrics (such as those used to provide capacity control in traditional statistical learning theory) and metrics based on fitting eigenvalue distributions to truncated power law (PL) distributions (which derive from statistical mechanics, in particular the recently-developed Theory of Heavy-Tailed Self Regularization, and which characterize the strength of correlations in a DNN). Using the publicly-available *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 find that norm-based metrics do a reasonably good job at predicting quality trends in well-trained models, i.e., they can discriminate amoing "good-better-best" models. On the other hand, for poorly trained models, i.e., to distinguish "good-versus-bad" models—which, arguably is the point of needing quality metrics—norm-based metrics can qualitatively fail. We also find that PL-based metrics do much better—quantitatively better at discriminating aming "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 models, e.g., understanding layer-wise *correlation flow*, and evaluate post-training modifications such as model distillation, increasing the data set size, and other model improvements.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

XXX, datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Charles H. Martin, Serena Peng, and Michael W. Mahoney. 2XXX. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data . In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122XXX

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 intelligence (AI), arises when the model *user* is not the model *developer*. Here, one may not have access to either the training data nor 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 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 marketing, are unsupervised and have no obvious labels for comparison and/or evaluation.

Most importantly, in industry, one faces unique practical problems such as: do we have enough data for this model? Indeed, high

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122XXX

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 coprus 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 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 matrics. 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 different. Unlike more common ML approaches, *we do not seek a bound on the generalization* (e.g., by evaluating training/test error during training), *we do not seek a new regularizer*, and *we do not aim to evaluate a single model* (e.g., hyperparmeter 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 aim to predict trends in the quality of pre-trained models without access to training or testing data.*

In more detail, our main contributions are the following.

- Motivated by practical problems, we formulate the question "Can one predict trends in the quality of state-of-the-art neural networks without access to training or testing data?"
- To answer this question, we analyze hundreds of publicly-available pretrained state-of-the-art models from CV and NLP.
- We find that norm-based metrics do a reasonably good job at predicting quality trends in well-trained models, i.e., they can be used to discriminate between "good-better-best" models, but they can qualitatively fail for models that may not be well-trained, i.e., to distinguish "good-versus-bad" models.
- We find that PL-based metrics do much better—quantitatively better at discriminating good-better-best, and qualitatively better at discriminating good-versus-bad models.
- We find that PL-based metrics can also be used to characterize fine-scale model properties, including layer-wise *correlation flow*, and evaluating post-training modifications such as model distillation, increasing the dataset size, and other model improvements.

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 indentify 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]; and, in order to make our results fully reproducible, we provide Jupyter notebooks in the github repo for this paper [**?**], and with fully reproducible Google colab notebooks for the larger calculations.[4] [michael: FIX THIS LINK TO: Google Colab OR THE REPO.]

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 4, 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 5, we present results based on an analysis of hundreds of pretraind DNN CV models, showing how well each metric predicts the reported test accuracies, and how the Alpha-Norm metric(s) perform remarkably well. Finally, in Section 6, we provide a brief discussion and conclusion.

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 work is, however, loosely related to several other lines of work.

Statistical mechanics theory for DNNs. Statistical mechanics ideas have long had influence on DNN theory and practice [2, 8, 10]; and our best-performing metrics (those using fitted PL exponents) are based on statistical mechanics [10–14], in particular the recently-developed *Theory of Heavy Tailed Self Regularization (HT-SR)* [11, 13, 14]. 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

[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.

[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.

[4] https://github.com/CalculatedContent/kdd2020 [michael: different name?][charles: suggestions?][michael: Anonymous, just for anonymized submission.]

more common approach are available [2, 8]. We *use* statistical mechanics in a broader sense, drawing upon results from quantitative finance and random matrix theory. Thus, much more relevant for our metholodogical approach is older work of Bouchaud, Potters, Sornette, and coworkers [4–6, 17] on the statisical 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 [3, 9, 16]. In this area, theoretical work aims to prove generalization bounds, and applied work uses these norms as a regularization function 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.

Practical problems need a practical theory. [Exiting ML theory does not address many practical AI problems that our techiques directly target.] Here are several.

- **Simplicity, or lack of, accuracy metrics.** [michael: XXX. Put in two or at most three sentences.]
- **Information leakage in the production pipeline.** [michael: XXX. Put in two or at most three sentences.]
- **Cost of acquring labeled data.** [michael: XXX. Put in two or at most three sentences.]

[charles: see comments in tex file][michael: Let's touch base, not sure what to do.]

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:

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

Assume we are given several pretrained DNNs, e.g., as part of an architecture series. The models have been trained on (unspecified and unavailable) labeled data $\{d_i, y_i\} \in \mathcal{D}$, using Backprop, by minimizing some (also unspecified and unavailable) loss function $\mathcal{L}(\cdot)$. We expect that most well-trained, production-quality models will employ one or more forms of on regularization, such as Batch Normalization, 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 layer weight matrices $\mathbf{W}_l$.

DNN Quality Metrics. Each DNN layer contains one or more layer 2D weight matrices, $\mathbf{W}_l$, and/or 2D feature maps, $\mathbf{W}_{i,l}$, extracted from 2D Convolutional layers. (For notational convenience, we may drop the i and/or i, l subscripts below; and we let $\mathbf{W}$ be a generic $N \times M$ weight matrix, with $N \geq M$.) We have examined a large number of possible quality metrics. The best performing metrics (recall that we can only consider metrics that do not use training/test data) depend on the norm and/or spectral properties of weight matrices, $\mathbf{W}$.[5] Consider the following.

- Frobenius Norm: $\|\mathbf{W}\|^2_F = \sum_{i,j} w^2_{i,j} = \sum_{i=1}^{M} \lambda^2_i$
- Spectral Norm: $\|\mathbf{W}\|_\infty = \lambda_{max}$ [michael: Should this be $\sqrt{\lambda_{max}}$?]
- Weighted Alpha: $\hat{\alpha} = \alpha \log \lambda_{max}$
- α-Norm (or α-Shatten Norm): $\|\mathbf{W}\|^\alpha_\alpha = \sum_{i=1}^{M} \lambda^\alpha_i$

The first two quantities are well-known in ML. The last two deserve special mention. For all these quantities, λ_i is the i^{th} eigenvalue of the *Empirical Correlation Matrix*, $\mathbf{X} = \mathbf{W}^T\mathbf{W}$, and λ_{max} is the maximum eigenvalue of $\mathbf{X}$. (These eigenvalues are the squares of the singular values σ_i of $\mathbf{W}$, i.e., $\lambda_i = \sigma^2_i$.) For the last two quantities, α is the PL exponent that arises in the recently-developed HT-SR Theory [11, 13, 14]. 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,

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

where λ_{max} *is* the largest eigenvalue of $\mathbf{X} = \mathbf{W}^T\mathbf{W}$. 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 to the appropriate power. Consider, e.g., the *α-Shatten Norm metric*,

$$\sum_l \log \|\mathbf{W}_l\|^{\alpha_l}_{\alpha_l} = \sum_l \alpha_l \log \|\mathbf{W}_l\|_{\alpha_l}. \quad (3)$$

Informallly, 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),

$$\mathcal{C} \sim \|\mathbf{W}_1\| \times \|\mathbf{W}_2\| \times \cdots \times \|\mathbf{W}_L\|, \quad (4)$$

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

$$\log \mathcal{C} \sim \log \|\mathbf{W}_1\| + \log \|\mathbf{W}_2\| + \cdots + \log \|\mathbf{W}_L\|, \quad (5)$$

takes the form of an Average Log Norm. For the *Frobenius norm metric* and *Spectral norm metric*, we can use Eqn. (5) directly. (When taking $\log \|\mathbf{W}_l\|^2_F$, the 2 comes down and out of the sum, and thus ignoring it only changes the metric by a constant factor.) For the α-Shatten Norm metric, however, α_l varies from layer to layer, and so in Eqn. (3) it cannot be taken out of the sum.

For the *Weighted Alpha metric*, we average of $\hat{\alpha}$ over all layers, i.e., $\sum_l \hat{\alpha}_l = \sum_l \alpha_l \log \lambda_{max,l}$. The Weighted Alpha metric was introduced previously [14], 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-Norm metric. One would expect $\hat{\alpha}$ to approximate the log α-Norm very well for $\alpha < 2$ and reasoably well for $\alpha \in [2, 5]$ [15].

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 of the amount of correlation in a DNN layer weight matrix $\mathbf{W}$. It is valid for $\lambda < \lambda_{max}$, 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.

[5] 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 that.

Series	#	$\log\langle\|\mathbf{W}\|_F\rangle$	$\log\langle\|\mathbf{W}\|_\infty\rangle$	$\hat{\alpha}$	$\log\langle\|\mathbf{X}\|_\alpha^\alpha\rangle$
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. VGG, ResNet, and DenseNet were pretrained on ImageNet, and ResNet-1K was pretrained on ImageNet-1K.

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.

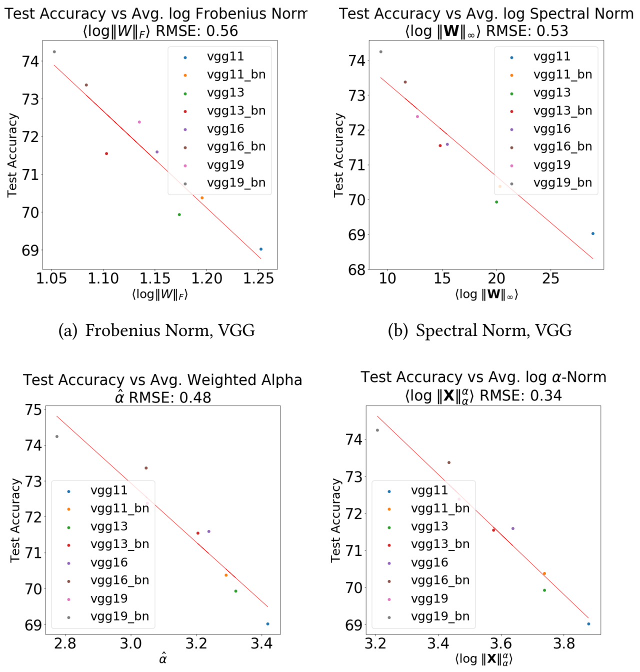
4 COMPARISON OF CV MODELS

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 [?] dataset, and each of which is distributed with the current opensource pyTorch framework (version 1.4) [?]. This also includes a larger set of ResNet models, trained on the ImageNet-1K dataset [?], provided on the OSMR "Sandbox for training convolutional networks for computer vision" [?], 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 Batch Normalization), and it plots the four quality metrics versus the reported Test accuracies [?], 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 generalziation (i.e., greater accuracy, lower error). While all four metrics perform well, notice that the Log α-Norm metric ($\log\|\mathbf{W}\|_\alpha^\alpha\rangle$) 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 [15], performs second best (with an RMSE of 0.48, see Table 1).

See Table 1 for a summary of results for Top1 Accuracies for all four metrics on 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 th e 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 "good-better-bes t" models, with PL-based metrics performing somewhat better.



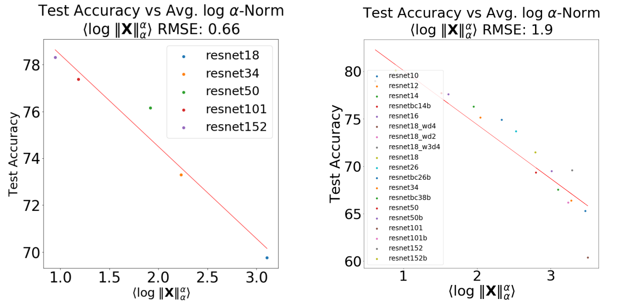
(a) Frobenius Norm, VGG

(b) Spectral Norm, VGG

(c) Weighted Alpha, VGG

(d) α-Norm, VGG

Figure 1: Comparison of average log Norm empirical quality metrics versus reported test accuracy for pretrained VGG models (with and without Batch Normalization (BN)), trained on ImageNet, available in pyTorch (v1.x). Metrics fit by linear regression, RMSE reported.



(a) ResNet, α-Norm

(b) ResNet-1K, α-Norm

Figure 2: Comparison of the avergage α-Norm empirical quality metric $\langle\log\|\mathrm{X}\|_\alpha^\alpha\rangle$ versus reported Top 1 reported Test Accuracy for the ResNet and ResNet-1K pretrained (pyTorch) models.

Notice that the DenseNet series, only has 4 data points. In our larger analysis, in Section **??** we only include series with 5 or more models. For completemnss and reproducibility here, the DenseNet Log α-Norm plot is given in the Appendix.

Variations in Data Set Size. We want to point out, in particular, how the empirical log Norm metrics vary with data set size. Figure 2 plots and compares the Log α-Norm metric–the best performing metric–for the full ResNet, trained on the full ImageNet dataset, against the ResNet-1K, which has been trained on a much smaller ImageNet-1K data set. Here, the Log α-Norm is much better that 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 an RMSE of 0.66, whereas the ResNet-1K series also shows good correlation, but has a much larger RMSE of 1.9. As expected, the higher quality data set shows a better fit, even with less data points.

Correlation Flow. We can learn much more about a pretrained model by going beyond average values 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, much more detailed set of plots is available at [?]; but note that the corresponding layer-wise plots for Frobenius and Spectral norms are much less interesting than the results we present here.)

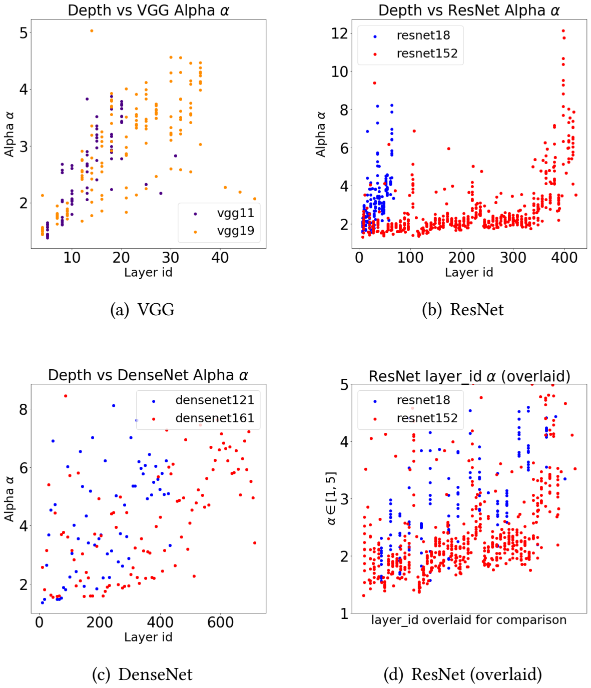
In the VGG models, Figure 3(a) shows that the PL exponent α systematically increases as we move down the network, from data to labels, in the Conv2D layers, starting with $\alpha \lesssim 2.0$ and reaching all the way to $\alpha \sim 5.0$; and then, in the last three, large, fully-connected (FC) layers, 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, 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 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 $\alpha \lesssim 8.0$) and much less systematic throughout the network.

[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-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 accurary with far fewer parameters (and GPU memory requirements), and ResNet models of up to 1000 layers have been trained.[?].] We conjuecture that the effeciency and effectivness 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 HT 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.

More generally, we can understand Figure 3(a) in terms of the *Correlation Flow*. Recall that the log α-Norm metric and the Weighted Alpha metric are based on HT-SR Theory [11, 13, 14], which is in turn based on ideas from the statisical mechanics of heavy tailed and



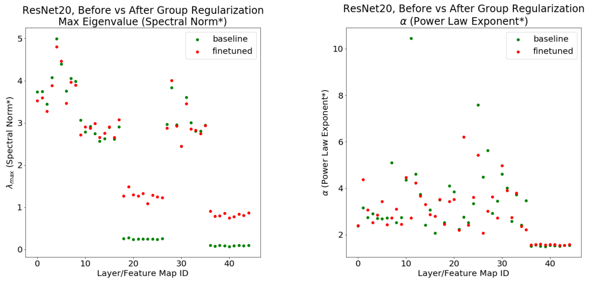
(a) VGG

(b) ResNet

(c) DenseNet

(d) ResNet (overlaid)

Figure 3: Correlation Flow: Power Law exponent α versus layer id, for the least and the most accurate models in VGG (a), ResNet (b), and DenseNet (c) series (VGG without BatchNorm, and the Y axes on each plot are different.) Figure (d) displays the ResNet models (b), zoomed in to *alpha* $\in [1, 5]$, and with the layer ids overlaid on the X-axis, to allow a more detailed analysis for the most strongly correlated layers. Notice that ResNet152 exhibits different and much more stable behavior across layers This contrasts with how both VGG models gradually worsen in deeper layers and how the DenseNet models are much more erratic.



(a) λ_{max} for ResNet20 layers

(b) α for ResNet20 layers

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

strongly correlated systems [4–6, 17]. 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) Power law (PL), with exponents $\alpha \in [2, 4]$. Much larger values ($\alpha \gg 5$ *or* 6) may relfect poorer PL fits, whereas smaller values ($\alpha \rightarrow 2$), are associated with models that generalize better. Informally, one would expect a DNN model to perform well when it fascilitates the propagation of information/features across layers. Previous studies argue this by computing the gradients over the input data. In the absence of training/test data, we can to try to quantify this by measuring the PL properties of empirical weight matrices. Smaller α values correspond to layers in which correlations across multiple scales are better captured [11, 17], and thus we expect that small values of α that are stable across multiple layers enable better *correlation flow* through the network.

Correlation Flow: PL vs. Norm Metrics. The similarity between norm-based metrics and α-based metrics suggests a question: is the Weighted Alpha metric just a variation of the more familiar norm-based metrics, or does it capture something different? More generally, do fitted α values contain information not captured by norms? To show that it is not just a variation and that α does capture something different, consider the following example, which looks at a compressed / distilled DNN model [7].

Scale Collapse; or, Distillation may break models. In examining hundreds of pretrained models, we have found several anamolies that demonsrate the power of our approach. Here, we demonstrate that some distillations 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.[6] We analyze the pre-trained 4D_regularized_5Lremoved baseline and finetuned models. The reported baseline test accuracies ($Top1 = 91.45$ and $Top5 = 99.75$) are better than the reported finetuned 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) suggests that the baseline Spectral Norms should be *smaller* on average than the finetuned ones should be smaller. *The oppopsite is observed.* Figure 4 presents the Spectral Norm (here denoteds $\log \lambda_{max}$) and PL exponent (α) for each individual layer weight matrix $\mathbf{W}$.[7] On the other hand, the α values do not differ systematically between the baseline and finetuned models. Also (not shown), the average (unweighted) baseline α is *smaller* than the finetuned average (as predicted by HT-SR Theory, the basis of $\hat{\alpha}$).

Note that Figure **??** depcits 2 very large $\alpha \gg 6$ for the baseline model, but not for the finetuned model. This suggests that the baseline model has at least 2 over-parameterized layers, and the distillation method does, in fact, improve the finetuned model by compressing these layers.s

[michael: XXX. WORK ON THIS PAR AFTER DO NLP EXAMPLES.] Our interpretation of this is the following. The pre-trained 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 we cosidered above. [charles: MOvE THIS SOMEWHERE ELSE AND CLEAN UP THIS SECTION Notice while norms make good regularizers for a single model, there is no reason *a priori* to expect them correlate so well with the test accuracies across different models. However, we do expect the PL α to do so because it effectively measures the amount of correlation in the model] The reason for this is that the `distiller` Group Regularization technique [spuriously increases] the norms of the $\mathbf{W}$ pre-activation maps for at least two of the Conv2D layers. [michael: This impedance shuffling or whatever it is called messes up norms, but the strong correlations are still preserved, as seen in the $\hat{\alpha}$ metric, and the model quality is good. CLARIFY.] [michael: XXX. INTERPRET IN TERMS OF XXX GOOD VERSUS BAD, AS OPPOSED TO GOOD BETTER BEST.]

sectionComparison of NLP Models

In this section, we examine quality metrics described in Section 3 for several NLP model architectures. Within the past two years, nearly 100 open source, pre-trained 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 matrics [**?**]. 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 α PL exponents. Based on HT-SR Theory, this suggests that these models fail to successfully capture 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, and thus, they provide a very different test for our empirical quality metrics.

[6] For details, see https://nervanasystems.github.io/distiller/#distiller-documentation and also https://github.com/NervanaSystems/distiller.

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

Where Norm-base metrics perform reasonably well on well-trained NLP models, they behave anomalously poorly-trained models. Indeed, in "bad models", the weight matrics may display rank collapse, decreased Frobenius mass, or unsually small Spectral norms, which may be mis-interpreted as "smaller is better." In contast, PL-based metrics (the log α-Norm metric, $\langle\log\|\mathbf{W}\|_{\alpha}^{\alpha}\rangle$, and the Weighted Alpha metric, $\hat{\alpha}=\alpha\log\lambda_{max}$) display consistant behavior even on poorly trained models, and we can use these metrics to help indentify where the archiectures need repair, and/or more and/or better data is needed.

What does large α mean? We do note, however, that many NLP models, such as the GPT and BERT, may have weight matrices with unusually large PL exponents ($\alpha \gg 6$), which indicates these matrices may be *under*-correlated (i.e over-parameterized), although the truncated PL fit itself may not be very reliable because the MLE estimator is unreliable in this range. Phenomenologically, if we examine the ESD visually, we can usually describe these $\mathbf{W}$ as in the *Bulk-Decay* or *Bulk-plus-Spikes* phase[**?**]. We have conjectured previously that very well-trained DNNs would not have many *outlier* $\alpha > 6$, and improved versions of GPT (shown below, and BERT (not shown) confirm this.

The differences between α and $\hat{\alpha}$. To avoid confusion, let us clarify the differences between α and $\hat{\alpha}$. We fit the ESD of the correlation matrix $\mathbf{X}$ to a truncated power law (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 $\lambda < \lambda_{max}$, and it scale invariant (does not depend on the normalization of $\mathbf{W}$ or $\mathbf{X}$.) The Weighted-Alpha metric, $\hat{\alpha}=\alpha\log\lambda_{max}$, approximates the log α Norm. $(\log\|\mathbf{X}\|_{\alpha}^{\alpha})$ (derived using statistical mechanics and RMT [(in progress)], and lets us compute a balanced, weighted average log PL norm metrix for the entire DNN. The $\hat{\alpha}$ metric also behaves like an imrpoved, weighted log average Spectral Norm, and may track this metric in some cases.

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 sizes of both the data set and architectures. 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 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 GPT to the well-trained GPT2, we can indentify empirical indidcators for when a model has in fact been poorly-trained and thus may perform poorly when deployed.

[charles: More details here.] The GPT models we analyze are deployed with the popular HuggingFace PyTorch library [**?**]. 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) matrics. 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 GPTand 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 include 12, 24, 36, and 48 layers, respectively, with increasingly larger weight matrices.

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

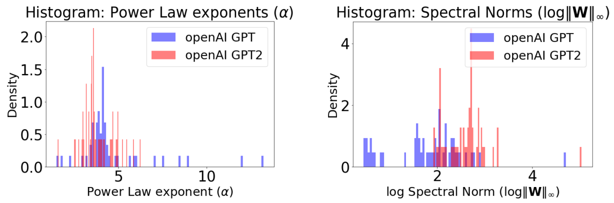
Table 2: Average value for the Average log Norm metrics and the Weighted Alpha metric for pretrainnd OpenAI GPT and GPT2 models. Column # refers to number of layers treated. Note averages do not include the first embedding layer(s) because they are not (implicitly) normalized.

Average Empirical 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 the PL exponents α for 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. Observe also that (with one minor exception involving the log Frobenius norm metric) all four metrics decrease as one goes from GPT2 medium to large to xl, indicating that the larger models indeed look better than the smaller models.

Figure 5(a) shows the histogram (empirical density), for all layers, of α for GPT (blue) and this GPT2 (red) model. These two histograms are very different, with GPT2 having both a notably smaller mean α, and far fewer unusually-large outlying α values.

From this, we see that α provides a good quality metric for comparing these two models, the"bad" GPT vs. the "good" GPT2. The deficient GPT has numerous unsually large α exponents–meaning they are not Heavy Tailed at all. Indeed, we expect that a poorly trained model lack Heavy Tailed behavior in all layers. On the other hand, as expected, the improved GPT2 model has, on average, smaller α than the older GPT, with all $\alpha \leq 6$.

Scale Collapse in Poorly Trained Models. However, as seen in Figure 5(b), the "bad" GPT model has, spuriously, many smaller Spectral Norms $(\log\|\mathbf{W}\|_\infty)$, compared to the "good" GPT2–which violates the belief that smaller Spectral Norms are always better. Indeed, because there are so many anonymously small $\|\mathbf{W}\|_\infty$, it appears that the GPT model may be exhibiting the *scale collapse* observed in the distilled CV models (above). This is an extremely important observation because it demonstrates that while the Spectral Norm may correlate well with predicted test error, it is not a good indicator of the overall quality of a model, and using it as an empirical metric may give spurious results when applied to poorly trained or otherwise deficient models.



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

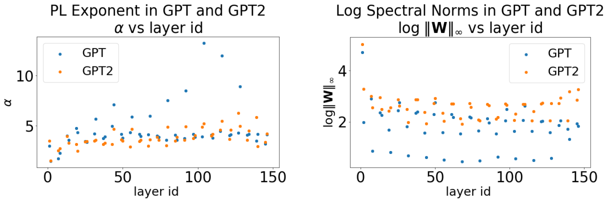
Figure 5: Comparison of PL exponents, α, and log Spectral Norms, ($\log \|\mathbf{W}\|_\infty$), for the OpenAI GPT and GPT2 (small) pretrained models.

Note that Figure 5(b) also shows some unusually large Spectral Norms, but, here, simply correspond to the change in normalization of the word embedding layer(s) and can be ignored. From Figure 6(b) (below), we see that these correspond to the first embedding layer(s). These layers appear to have a different effective normalization, and therefore a different scale. [charles: We discuss this further in the Appendix] Here, we do not include them in our computed average metrics in Table 2, and do not include them in the histogram plot in Figure 5(b).

Correlation Flow in GPT and GPT2. The correlation flow (α as a function of depth / layer id) also differs significantly between GPT and GPT2. Figure 6(a) plots α vs the depth (i.e. a 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 model shows consistant 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.

Notice also that the Spectral Norms (Figure 6(b)) and the $\alpha-$Norms (Figure ??) display an unusually some mall values for GPT, and some increasing trending values for GPT2. Generally speaking, we do not think scale-dependennt norm metrics can be directly applied to distinguish "good" vs "bad" models because of the anamalous scale collapse that is frequently observed in bad models.

GPT2: medium, large, xl. We now look across the series of increasingly improving GPT2 models, (i.e. "good, better, best"), by examining both the PL exponent α, as well as the various log Norm metrics. In general, as we move from GPT2-small to GPT2-xl, the histograms for both α exponents and the log Norm metrics downshift from larger to smaller values. See Figure 7, which shows the histograms over the layer weight matrics for fitted α, and the log Spectral Norm ($\log \|\mathbf{W}\|_\infty$) and log Alpha Norm ($\log \|\mathbf{W}\|_\alpha^\alpha$) metrics.

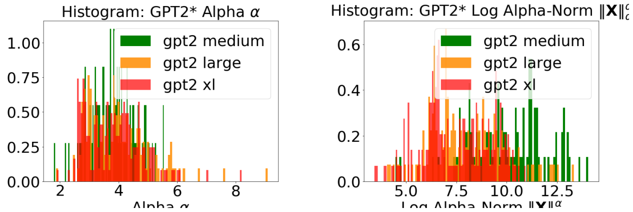


(a) Power Law exponent (α) (b) Spectral Norm ($\log \|\mathbf{W}\|_\infty$)

Figure 6: Comparison of Correlation Flow and Spectral Norm for OpenAI GPT and GPT2

We see that the average α decreases with increasing model size, although the differences are less noticible between the differing "good, better, best" GTP2 models than between the "good vs bad" GPT2 and GPT models. Unlike GPT, however, the layer (log) Spectral Norms ($\log |\mathbf{W}\|_\infty$) and (log) Alpha Norms ($\log |\mathbf{W}\|_\alpha^\alpha$) behave more as expected for GPT2 layers, with the larger models, consistently having smaller norms. Likewise, Figure **??** shows decreasing average log Spectral Norms with the larger models. 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 is getting smaller, towards 2.0, the tails of the distribution shifts right, with larger GPT2 models having more usually large α. We suspect this indicates that these larger GPT2 models are over-parameterized/under-optimized and that they have capacity to support datasets even larger than the recent XL 1.5B release [**?**].



(a) PL exponent (α) (b) log Alpha Norm

Figure 7: PL exponents (α), log Spectral norm ($\log \|\mathrm{W}\|_\infty$), and log Alpha norm ($\log \|\mathrm{X}\|_\alpha^\alpha$) for different size models in the GPT2 architecture series. (Plots omit the first 2 (embedding) layers, because they are normalized differently giving anamolously large values.)

5 COMPARISON OF ALL 168 CV MODELS

[michael: Now we go back to the CV models and look at over 100 of them, and draw' more broad conclusions about why alpha and the new norm is better, by looking at the MSE]

Here, we use the Weightwatcher tool to analyze over 100 pretrained computer vision (CV) models Pytorch, including image classification and segmentation models. They have been pretrained on nine datasets, ImageNet-1K, and CIFAR-10, CIFAR-100, Street

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 metrics across 5 computer vision datasets, 17 Architectures, covering 168 (out of 309) different pretrained DNNs. We only conclude regressions for architecturs with 4 or more data points, and which are postively correlated with the test error. These results can be readily reproduced using the Google Colab notebooks (see Appendix)

View House Numbers (SVHN), Caltech-UCSD Birds-200-2011 (CUB-200-2011), Pascal VOC2012, ADE20K, Cityscapes, and Common Objects in Context (COCO). The pretrained models and their accuracy metrics are summarized in the osmr github [charles: add link]

We provide more details in Appendix A.

In this paper, we propose that the Weightwatcher tool could be used to predict the trends in the generalization accuracy of deep neutral network without a test set. To test our proposition, we choose simple linear regression to analyze the relationship between the Weightwatcher metrics and the traditional accuracy metric obtained with a test set. We regress the metrics on the Top1 (and Top5) reported errors (as dependent variables). These include the Top5 errors for the ImageNet-1K mode, the percent error for the CIFAR-10/100, SVHN, CUB-200-2011 models, and the Pixel accuracy (Pix.Acc.) and Intersection-Over-Union (mIOU) for XXX. We regress them individually on each of the Weightwatcher log Norm metrics, as described above.

6 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 *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), and 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 statisical mechanics of heavy tailed and strongly correlated systems [4–6, 17], 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), but again, our results are the first to demonstrate this. It is gratifying, also, that PL exponents can be used to provide fine-scale insight such as rationalizing the flow of correlations through a network.

We conclude with a few thoughts on what a *practical theory* of DNNs should look like. We distinguish between what we will call a *phenomenological theory* (that describes empirical relationship of phenomena to each other, in a way which is consistent with fundamental theory, but is not directly derived from that theory) and what can be called a *first principles theory* (that is applicable to toy models, but that does not scale up to realistic systems if one includes realistic apsects of realistic systems). For most complex highly-engineered systems (aside from complex AI/ML systems), one *uses* phenomenological theory rather than first principles theory. (One does not try to solve the Schrödinger equation if one is interested in building a bridge or flying an airplane.) Our results, which are based on our *use* of sophisticated statistical mechanics theory to solve an important practical DNN problems, suggests that this approach should be of interest more generally for those interested in developing a practical DNN theory. [charles: see latex file] [michael: Let's touch base to get this right.]

ACKNOWLEDGMENTS

MWM would like to acknowledge ARO, DARPA, NSF, and ONR 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] 2018. WeightWatcher. https://pypi.org/project/WeightWatcher/.

[2] Y. Bahri, J. Kadmon, J. Pennington, S. Schoenholz, J. Sohl-Dickstein, and S. Ganguli. 2020. Statistical Mechanics of Deep Learning. *Annual Review of Condensed Matter Physics* (2020), 000–000.

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

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

[5] J. P. Bouchaud and M. Potters. 2011. Financial Applications of Random Matrix Theory: a short review. In *The Oxford Handbook of Random Matrix Theory*, G. Akemann, J. Baik, and P. Di Francesco (Eds.). Oxford University Press.

[6] J. Bun, J.-P. Bouchaud, and M. Potters. 2017. Cleaning large Correlation Matrices: tools from Random Matrix Theory. *Physics Reports* 666 (2017), 1–109.

[7] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. 2017. *A Survey of Model Compression and Acceleration for Deep Neural Networks.* Technical Report Preprint: arXiv:1710.09282.

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

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

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

[11] C. H. Martin and M. W. Mahoney. 2018. *Implicit Self-Regularization in Deep Neural Networks: Evidence from Random Matrix Theory and Implications for Learning.* Technical Report Preprint: arXiv:1810.01075.

[12] C. H. Martin and M. W. Mahoney. 2019. Statistical Mechanics Methods for Discovering Knowledge from Modern Production Quality Neural Networks. In *Proceedings of the 25th Annual ACM SIGKDD Conference.* 3239–3240.

[13] C. H. Martin and M. W. Mahoney. 2019. Traditional and Heavy-Tailed Self Regularization in Neural Network Models. In *Proceedings of the 36th International Conference on Machine Learning.* 4284–4293.

[14] C. H. Martin and M. W. Mahoney. 2020. 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.*

[15] C. H. Martin and M. W. Mahoney. 2020. Unpublished Results.

[16] B. Neyshabur, R. Tomioka, and N. Srebro. 2015. Norm-Based Capacity Control in Neural Networks. In *Proceedings of the 28th Annual Conference on Learning Theory.* 1376–1401.

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

A APPENDIX

In this appendix, we provide more details on several issues that are important for reproducibility of our results.

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 $w = h = k$, giving $(k \times k)$ 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 $((k \times k \times out) \times in)$, 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 coeffients [?], 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 $1/N$ 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 [?], 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 $latex \frac{1}{\sqrt{N}}$ normalization present in other layers. For example, in GPT, most layers, the maximum eigenvalue $\lambda_{max} \sim O(10-100)$, but in the first embedding layer, the maximum is of order N (the number of words in the embedding), or $\lambda_{max} \sim O(10^5)$. For GPT and GPT2, we treat all layers as-is (although one may to normalize the first 2 layers by $\mathbf{X}$ by $\frac{1}{N}$, or to treat them as an outlier).

A.2 Reproducing Sections 4 and 5

We provide a github repository for this paper that includes jupyter notsbooks that fully reproduce all results. All results have been produced using the weightwatcher tool (v0.2.7). The ImageNet and OpenAI GPT pretrained models are provided in the current pyTorch, torchvision, and huggingface distributions, as specified in the `requirements.txt` file.

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
5	WeightWatcher-OpenAI-GPT.ipynb
6, 7	WeightWatcher-OpenAI-GPT2.ipynb

Table 4: Jupyter notebooks used to reproduce Tables 1 and 2, and Figures 1,2,3,5,6, and 7.

A.3 Reproducing Section 3, Distiller Model

A.4 Reproducing Table 3, Section 6

We provide several Google Colab notebooks which can be used to reproduce the result in Table 3. 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. [8] The data can be generated in parallel by running each Colab notebook simultaneously on the same account, The final results are generated with [charles: move notebooks to repo and finish]

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, Pascal-VOC2012, ADE20K, and COCO datasets as there are less than 15 models for those datasets. 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 further explain how to reproduce our analysis, we run three batches of linear regressions. First at the global level, we divide models by datasets and run regression 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, you could still see that most plots show positive relationship with positive coefficients The regressions are shown in Figure 8.

To generate the results in Table 3, we run linear for each architecture series in Table 6, regressing each empirical log norm metric against the reported Top 1 (and Top 5) errors (as listed on the `osmr/imgclsmob` github repository README file. We filter out regressions with less than five datapoints. We record the R-squared and mean squared errors (MSE). The final results for all models is provided in the `XXXXXX`.

A.5 XXX: PLACEHOLDER STUFF PROBABLY TO BE REMOVED

Some other comments that we need to weave into a narrative eventually after later sections are written:

- GPT versus GPT2. What happens when we don't have enough data? This is the main question, and we can use out metrics

[8] https://github.com/osmr/imgclsmob

Dataset	# of Models
imagenet-1k	78
svhn	30
cifar-100	30
cifar-10	18
cub-200-2011	12

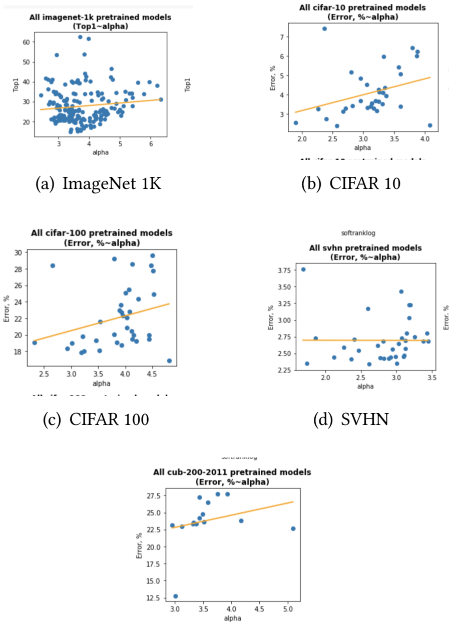
Table 5: Datasets used

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	6
SqueezeNet/SqueezeResNet	6

Table 6: Architectures used

to evaluate that, but we also get very different results for GPT versus GPT2.

- The spectral norm is a regularizer, used to distinguish good-better-best, not a quality metric. For example, it can "collapse," and for bad models we can have small spectral norm. So, it isn't really a quality metric.
- One question that isn't obvious is whether regularization metrics can be used as quality metrics. One might think so, but the answer isn't obviously yes. We show that the answer is No. A regularizer is designed to select a unique solution from a non-unique good-better-best. Quality metrics can also distinguish good versus bad.
- (We should at least mention this is like the statistical thing where we evaluate which model is better, as oposed to asking if a given model is good, I forget the name of that.)
- There are cases where the model is bad but regularization metric doesn't tell you that. Quality should be correlated in an empirical way. Correlated with good-better-best; but also tell good-bad.
- Question: why not use regularier for quality? Answer: A regularizer selects from a given set of degenerate models one which is nice or unique. It doesn't tell good versus bad, i.e., whether that model class is any good.
- Thus, it isn't obvious that norm-based metrics should do well, and they don't in general.



(a) ImageNet 1K

(b) CIFAR 10

(c) CIFAR 100

(d) SVHN

(e) CUB 200

Figure 8: [charles: Preliminary charts:] **PL exponent α vs. reported Top1 Test Accuracies for pretrained DNNs available**[charles: ref] **for 5 different data sets.**

- We give examples of all of these: bad data; defective data; and distill models in a bad way. (Of course, bad data means bad model, at least indirectly, since the quality of the data affects the properties of the model.)
- We can select a model and change it, i.e., we don't just do hyperparameter fiddling.