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

Charles H. Martin* Serena Peng† Michael W. Mahoney‡

Abstract

In many applications, one must work with models that have been trained by someone else. For such *pretrained models*, one typically does not have access to training data or test data, and one 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 metrics drawn from traditional statistical learning theory (e.g., norm-based capacity control metrics) as well as metrics (e.g., fitted power law metrics used to characterize the degree of strong correlations in trained models) derived from the recently-developed Theory of Heavy-Tailed Self Regularization (HT-SR). 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 and natural language processing. 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." On the other hand, for models that may not be well-trained (which, arguably is the point of needing metrics to evaluate the quality of pretrained models), i.e., when we want to distinguish "good-bad," norm-based metrics can qualitatively fail. We also find that HT-SR metrics do much better, quantitatively better for at discriminating good-better-best and qualitatively better at discriminating good-bad. HT-SR 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 and model improvement.

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 and/or testing error. There are many problems with this approach. Well-known problems with just examining training/testing curves include that they give very limited insight into the overall properties of the model, they do not take into account the (often extremely 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 somewhat less well-known problem, but one that is increasingly important (in particular in industrial-scale ML—where the *users* of models are not the *developers* of the models) is that one may access to neither the training data nor the testing data. Instead, one may simply be given a model that has already been trained—we will call such an already-trained model a *pretrained model*—and be told to use it.

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

†XXX

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

Naïvely—but in our experience commonly, among both 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 ML 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 still other groups will use those models. The users of models can not be expected to know the precise 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. Having metrics to evaluate the quality of a ML model in the absence of training and testing data and without any detailed knowledge of the training and testing process—indeed, having theory for pretrained models, to predict how, when, and why such models can be expected to perform well or poorly—is clearly of interest.

In this paper, we present and evaluate quality metrics for pretrained neural network (NN) and deep neural network (DNN) models at scale.[1] To do so, 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).[2] These provide a large corpus of models that by some community standard are state-of-the-art.[3] [michael: XXX. MORE DETAILS.] Importantly, for all of these models, we have no access to training data or testing data.

[michael: XXX. LIST PLACES WHERE THEY ARE AVAIALBLE, HERE OR IN NEXT SECTION.]

XXX. To do this, we will compute a variety of *quality metrics* based on the spectral properties of the layer weight matrics. Note that unlike traditional ML approaches, however, *we do not seek a bound on the generalization* (e.g., by evaluating training/test error during training), and *we do not aim to evaluate a single model* (e.g., by training with differing hyperparmeters). Instead, we want to examine different models a 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.

- XXX TECHNICAL THING 1
- XXX TECHNICAL THING 2
- XXX TECHNICAL THING 3
- XXX TECHNICAL THING 4

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

[1] We reiterate: One could 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 NN models.

[2] 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.

[3] 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.

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 pretraind DNN CV models, showing how well each metric is correlated with the reported test accuracies, and how the Alpha-Norm metric(s) perform remarkably well. Finally, in Section 7, we provide a brief discussion and conclusion.

2 Background and Related Work

To our knowledge, there is very little work on the particular question we are addressing: namely, how to predict, in a theoretically-principled manner, the quality of large-scale state-of-the-art NNs, and 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, and we briefly discuss them here.

Statistical mechanical theory of NNs. XXX. Cite our stuff: [**?**], [**?**], [**?**], [**?**] [**?**], [**?**], [**?**]. Cite also Ganguli review and maybe other stuff.

XXX. 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 tiny things but has no hope of scaling up). [4]

Norm-based capacity control theory. XXX. MOST IRRELEVANT, BUT LIAO AND OUR SDM ARE RELATED. XXX. MAYBE BEAT ON INFINITELY WIDE OR SOMETHING ELSE.

Practical problems poorly addressed by theory. There are many very practical problems in ML that are poorly addressed by existing theory and that either motivated our work or should be addressable by our techniques. Here are several.

- **Lack of accuracy metrics.**
- **Lack of data.**
- XXX. LACK OF SOMETHING ELSE.

Importantly, there are many examples in ML where (as a practical matter) there is no reliable notion of accuracy, e.g., when generating fake text, when developing self driving car systems, and when distilling a reliable model in some way to obtain comparable training/test quality but that damages the model in some other subtle way. [michael: That last example is awkward. It would be good to have a better example and plant seeds for model distillation elsewhere.]

3 Methods

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

[4] In most areas where there are complex highly-engineered systems (beyond complex AI/ML systems), one used phenomenological theory rather than first principles theory. For example, one does not try to solve the Schrödinger equation if one is interested in building a bridge or an airplane.

$\mathbf{W}_l$ and biases $\mathbf{b}_l$, as:

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

We assume we are given several pretrained Deep Neural Networks (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 $\mathcal{L}()$. 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 weight matrices.

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.) [michael: XXX. PROBABLY WE DO NOT WANT THE FOLLOWING. INSTEAD, JUST PRESENT METRICS AND POINT TO PREVIOUS PAPERS FOR JUSTIFICATION. E.G., WE DONT WANT TO JUSTIFY THE IMPLICIT STATISTICAL INDEPENDENCE ASSUMPTION HERE. IF OKAY, JUST DELETE THIS RED BLOCK. We assume the layer weight matrices are statistically independent, allowing us to estimate the Complexity $\mathcal{C}$, or test accuracy, with a standard Product Norm, which resembles a data dependent VC complexity

$$\mathcal{C} \sim \|\mathbf{W}_1\| \times \|\mathbf{W}_2\| \cdots \|\mathbf{W}_L\|, \quad (2)$$

where $\mathbf{W}$ is an $(N \times M)$ weight matrix, with $N \geq M$, and $\|\mathbf{W}\|$ is a matrix norm. We will actually compute the log Complexity, which takes the form of an Average Log Norm:

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

] 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] We consider the following metrics.

- Frobenius Norm: $\|\mathbf{W}\|_F^2 = \|\mathbf{W}\|_2^2 = \sum_{i,j} w_{i,j}^2 = \sum_{i=1}^{M} \lambda_i^2$ [michael: Do we use norm or log norm, here and with other norms. Worth being very explicit and very consistent.]
- Spectral Norm: $\|\mathbf{W}\|_\infty = \lambda_{max}$ [michael: We should probably call the spectral norm $\|\mathbf{W}\|_2$, or change Frob norm notation and have an explicit remork. Let's decide.]
- Weighted Alpha Metric: $\hat{\alpha} = \alpha \log \lambda_{max}$
- α-Norm (or α-Shatten Norm): $\|\mathbf{X}\|_\alpha^\alpha = \sum_{i=1}^{M} \lambda_i^\alpha$, [michael: Be clear about how we average this across layers.]

The first two metrics are well-known in ML. The last two deserve special mention. For all these metrics, λ_i is the i^{th} eigenvalue of the *Empirical Correlation Matrix*, $\mathbf{X} = \mathbf{W}^T\mathbf{W}$, and so 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., $\lambda_i = \sigma_i^2$.) For the last two metrics, the exponent α is the power law exponent that arises in the recently-developed *Theory of Heavy Tailed Self Regularization (HT-SR)* [**?**, **?**, **?**]. Operationally, α is determined by using the publicly-available *WeightWatcher* tool ([**?**]) to fit

[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

the Empirical Spectral Density (ESD) of $\mathbf{X}$, i.e., a histogram of the eigenvalues, call it $\rho(\lambda)$, to a truncated power law

$$\rho(\lambda) \sim \lambda^{\alpha}, \quad \lambda \leq \lambda_{max}. \tag{3}$$

[michael: We need to be clear that this is a truncated power law fit and that λ_{max} comes from that and not the largest empirical eigenvalue.] The Weighted Alpha Metric was introduced previously [**?**], where (on a much smaller set of data than we consider here) it was shown to correlate well with the trends in reported test accuracies of pretrained DNNs. Based on this, here 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 $\alpha \in [2, 5]$ [**?**].

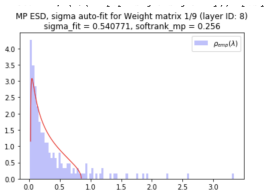
[michael: Need to say that such large α values don't mean much.]

[michael: Need to highlight difference between α and $\hat{\alpha}$.]

Spectral Analysis of Convolutional 2D Layers. There is some ambiguity in performing spectral analysis on Convolutional 2D (Conv2D) layers. A Conv2D layer can be represented as a 4-index tensor of dimension (w, h, in_ch, out_ch), specified by an $(w \times h)$ filter (or kernel) and in_ch / out_ch input / output channels, respectively (usually $in\_ch \leq out\_ch$). Typically, $w = h = k$, giving $(k \times k)$ tensor slices, or *pre-Activation Maps* $\mathbf{W}_{i,L}$ of dimension $(in\_ch \times out\_ch)$ each. There are at least three different approaches that have been advocated for applying the Singular Values Decomposition (SVD) to an Conv2D layer: run an SVD on each of the pre-Activation Maps $\mathbf{W}_{i,L}$, yielding $(k \times k)$ sets of M singular values; stack the feature maps into a single rectangular matrix of, say, dimension $((k \times k \times out\_ch) \times in\_ch)$, yielding in_ch singular values; compute the 2D Fourier Transform (FFT) for each of the (in_ch, out_ch) pairs, and run SVD on the resulting Fourier coeffients [**?**], leading to $\sim (k \times in\_ch \times out\_ch)$ non-zero singular values. Each method has tradeoffs. In principle, the third method is mathematically sound, but it is computationally expensive. For our analysis, because we are performing tens of thousands of calculations, we select the first method, which is numerically the fastest and is easiest to reproduce.[6]

XXX. XXX. THIS VERIFICATION IS NOT ABOUT CONV LAYERS, IT IS ABOUT PL MORE GENERALLY, CORRECT? WE SHOULD CLARIFY AND SQUISH. To verify that our approach is meaningful, we need to confirm that the ESD is neither due to a random matrix, nor due to unsually large matrix elements, but, in fact, captures correlations learned from the data. We examine typical layer for the pretained AlexNet model (distributed with pyTorch). Figure 1(a) displays the ESD for the first slice (or matrix $\mathbf{W}$) of the third Conv2D layer, extracted from a 4-index Tensor of shape $(384, 192, 3, 3)$. The red line displays the best fit to a random matrix, using the Marchenko pastur theory [**?**]. We can see the random matrix model does not describe the ESD very well. For comparison, Figure 1(b) shows the ESD of the same matrix, randomly shuffled; here looks similar to the red line plot of the orginal ESD. In fact, the empircal ESD is better modeled with a truncated power law distribtion. [michael: We may want to give a one-sentence summary of this par and fig at the end of the previous par.]

[6] We provide a Google Colab notebook where all results can be reproduced, with the option to redo the calculations with the third option for the SVD of the Conv2D.



(a) Actual ESD

(b) ESD of randomly shuffled matrix

Figure 1: ESD of AlexNet Conv2D pre-Activation map for Layer 3 Slice 1, actual and randomized. [michael: Need better figs here.]

Although the ESD is *Heavy Tailed*, this does not imply that the orginal matrix $\mathbf{W}$ is itself heavy tailed–only the correlation matrix $\mathbf{X}$ is. If $\mathbf{W}$ was, then it would contain 1 or more unusually large matrix elements, and they would dominate the ESD. Of course the randomized $\mathbf{W}$ would also be heavy tailed, but its ESD neither resembles the original nor is it heavy tailed. So we can rule out $\mathbf{W}$ being heavy tailed. [michael: These comments seem out of place, since they hold more generally than for the Conv2D layers.]

These plots tell us that the pre-activation maps of the Conv2D contains significant correlations learned from the data. By modeling the ESD with a power law distribution λ^{α}, we can characterize the amount of correlation learned; the smaller the exponent α, the more correlation in the weight matrix. [michael: These comments seem out of place, since they hold more generally than for the Conv2D layers.]

Normalization of Empirical Matrices. Normalization is an important, if underappreciated, practical issue. Importantly, the normalization of weight matrices does *not* affect the Power Law fits because the Heavy Tailed exponent α (as well as other metrics such as the Stable Rank and MP Soft Rank [**?**, **?**]) is scale-invariant. Norm-based metrics, however, do depend strongly on the scale of the weight matrix. Typically, to apply RMT, we would usually define Correlation Matrix with $1/N$ normalization and assume that the variance of $\mathbf{X}$ is either unity or a known constant.[7] Pretrained DNNs are typically initialized with random weight matrices $\mathbf{W}_0$, with the variance already normalized to $\sqrt{1/N}$, or some variant of this, e.g., the Glorot/Xavier normalization [**?**], or a $\sqrt{2/Nk^2}$ normalization for Convolutional 2D Layers. We do not have conrol over the final empirical normalization of these models; and we do *not* normalize (or renormalize) the Empirical Correlation Matrices, i.e., we use them as-is. The only exception to this 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 layers.

[michael: MOVE TO LATER: COMMENT ON HOW LOG NORM first and last layers behave, maybe somewhere else.]

[michael: CMOVE TO LATER: OMMENT ON HOW LOG PORM for GPT includes unusually high alpha, not meaningful other than to show the trend.]

The WeightWatcher Tool. We compute the metrics using the WeightWatcher tool (version 0.2.7), and we provide Jupyter notebooks in the github repo for this paper [**?**], making the results fully reproducible.

[michael: XXX. FEW SENTENCES ABOUT REPRODUCIBILITY MORE GENERALLY HERE.]

[7] For formal proofs of Heavy Tailed results, one typically needs a different normalization such as $1/N^{1-\alpha}$.

4 Comparison across series of CV models

In this section, we examine the 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 results for 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 use our quality metrics to understand the gross properties of these models, comparing and contrasting the three (of four, if ResNet-1K is a distinct series) model series. We also use our metrics to understand more fine-scale properties of these models as a function of model depth. [michael: And mention distillation, if we include that in this section.]

Quality Metrics versus Reported Test Accuracies. We have examined the performance of the four quality (Frobenius norm, Spectral norm, Weighted Alpha, and α-Norm) applied to each of the VGG, ResNet, ResNet-1K, and DenseNet series. To start, Figure 2 plots the four quality metrics versus the reported Test accuracies [**?**], as well as a basic linear regression line, for the pretrained models VGG11, VGG13, VGG16, and VGG19, with and without BatchNormalization. [michael: These are all log norms, right?] [michael: XXX. WHERE IS RMSE. IN THAT TABLE?] 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). Similar results (not shown) are obtained for the Top5 Accuracies. Notice that the log α-Norm metric $\log \|\mathbf{W}\|)_{\infty}^{\infty}$ [michael: XXX IS THAT A TYPO] performs best, with an RMSE of 0.34; and the second best metric is $\hat{\alpha}$, which is an approximate log α-Norm metric. [michael: Are those numbers stale, I don't see them, and some things seem in the other order.] [michael: We should probably point to the table if we cite quantitative results.]

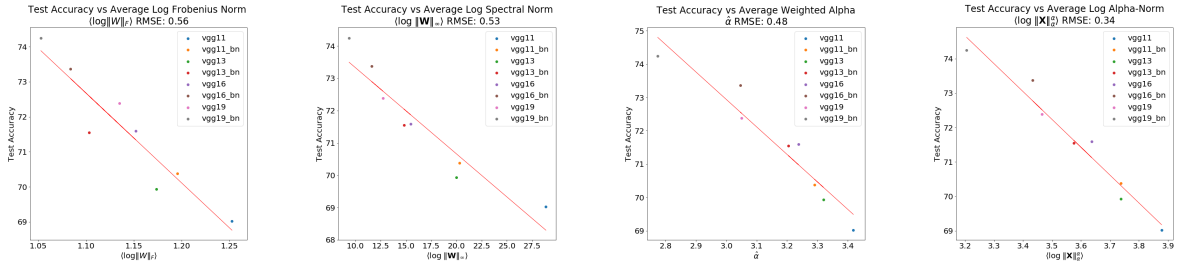
Figure 3 plots the best performing metric (α-Norm) for the full ResNet, ResNet-1K, and DenseNet series. (A much more detailed set of plots—including the behavior of all four metrods on each of the series—are availale at [michael: give web link].) For a more visual depiction, Figure 3 displays the log α-Norm metric $\log \|\mathbf{W}\|)_{\infty}^{\infty}$ versus test accuracies for the ResNet, ResNet-1K, and DenseNet series. Notice that ResNet series, which has been trained on the full ImageNet dataset, has an RMSE of 0.66, whereas the ResNet-1K series, which has been trained on the much smaller ImageNet-1K dataset, also correlated wells, but has a much larger RMSE of 1.9.

Table 1 summarizes results for all four metrics (Frobenius norm, Spectral norm, Weighted Alpha, and α-Norm) applied to the VGG, ResNet, ResNet-1K, and DenseNet series. [michael: XXX. AGAIN, WHERE IS RMSE, IN THAT TABLE?]

For all series, all four metrics correlate quite well with the reported Top1 test accuracies. Moreover, the α-Norm and/or the approximate Weighted Alpha metrics perform best.

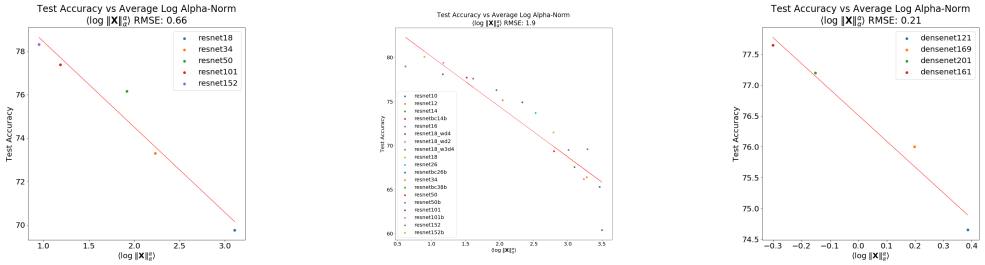
Correlation Flow in CV Models. We can learn much more about a pretrained model by going beyond average values to examining quality metrics for each layer weight matrix, $\mathbf{W}$, as a function of depth (or layer id). The most interesting results are seen when we plot the PL exponent, α, as a function of depth. See Figure 4, which plots α for each layer (the first corresponds to data, the last to labels) for the least accurate (shallowest) and most accurate (deepest) model in each of the VGG (without BatchNormalization), ResNet-1K, and DenseNet series.

In the VGG models, 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



(a) Frobenius Norm (b) Spectral Norm (c) Weighted Alpha (d) Alpha-Norm

Figure 2: Comparison of quality metrics versus reported test accuracy for pretrained VGG models, trained on ImageNet, available in pyTorch. XXX Plots will be updated and replaced. [michael: Figures are unreadable, can we make fonts bigger, etc.]



(a) ResNet (b) ResNet-1K (c) DenseNet

Figure 3: α-Norm versus reported Top1 Error for ResNet, ResNet-1K, and DenseNet models.

figure), indicating that the main effect of increasing depth is to increase the range over which α can increase, thus leasing 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, α also increases in the last few layers, but as the models get deeper, there is a wide range over which α values tend to remain small. [michael: Can we say something about exceptions which are larger than for VGG or DenseNet.] 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.

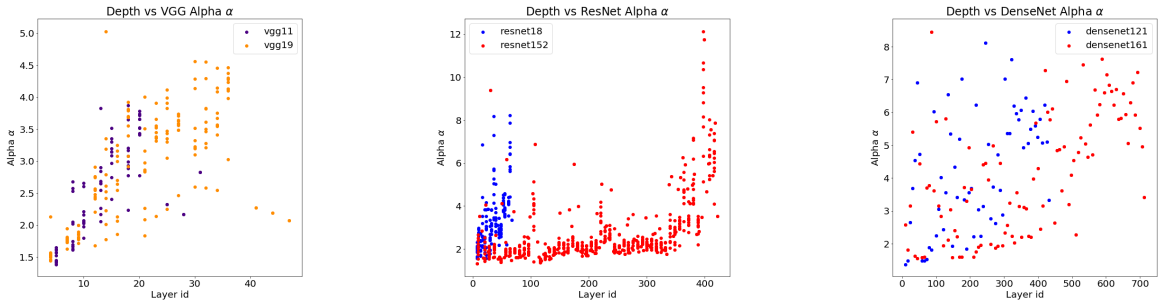
For the DenseNet models, α tends to increase as the layer id increases, in particular for layers toward the end. This is similar to what is seen in the VGG models, but with the DenseNet models, the variance is much larger (in particular for the earlier and middle layers, where it can range all the way to $\alpha \lesssim 8.0$) and much less systematic.

We can understand Figure 4 in terms of the *flow of correlation*, as follows. Informally, one would expect a DNN model to perform well when it fascilitates the propagation of information/features across layers. In the absence of training/test data, we can to try to quantify this by measuring the PL properties of weight matrices. Smaller α values correspond to layers in which correlations across multiple scales are better captured [**?**, **?**], and thus we expect that values of α that are stable across multiple layers enable better *correlation flow* through the network.

Consider the differences—in particular, the number of residual connections—between the VGG, ResNet, and DenseNet architectures. VGG, while a good model in many ways, was limited by needing a massive FC layers at the end of the model, requiring significant memory and com-

Series	#Models	Frobenius Norm $\|\mathbf{W}\|_F$	Spectral Norm $\|\mathbf{W}\|_2$	Weighted Alpha $\hat{\alpha} = \alpha \log \lambda_{max}$	Alpha-Norm $\|\mathbf{X}\|_\alpha^\alpha$
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 for linear fits (smaller is better) of quality metrics to Reported Top1 Test Error ([**?**][michael: Give refs again.]) for all pretrained models in the architecture series. VGG, ResNet, and DenseNet were pretrained on ImageNet, and ResNet-1K was pretrained on ImageNet-1K. [michael: Do we want Frob norm squared.]



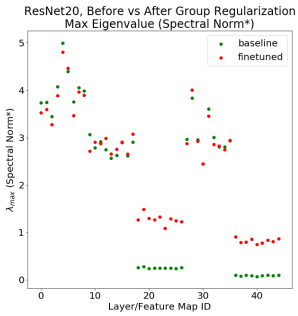
(a) VGG (b) ResNet-1K (c) DenseNet

Figure 4: PL exponent α versus layer for VGG, ResNet-1K, and DenseNet models. Y axes on each plot are different. ResNet-1K exhibits very different and much more stable behavior, which we interpret in the text as a *correlation flow* across the layers.

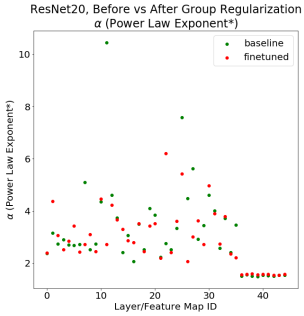
putational resources, and it's poor conditioning led to problems with vanishing gradients. The empirical manifestations of this (on weight matrices) are that fitted α values get much worse for deeper layers. ResNet greatly improved on VGG by introducing residual connections, allowing for greater accurary with far fewer parameters. (Indeed, 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. This should also be contrasted with the DenseNet model, 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 the correlation flow across layers, and leaving many layers very poorly optimized.

Behavior on Less Thoroughy Studied CV Models. Figure 2 and Table 1 show that—for well-trained and widely-studied models—the weighted $\hat{\alpha}$ metric performs better than but similar to norm-based metrics. This suggests an obvious question: is the $\hat{\alpha}$ metric just a variation of these more familiar norm-based metrics, or does it capture something different? To show that it is not just a variation and that it does capture something different, consider the following less thoroughy studied model, a compressed DNN model [**?**], where we observed that the average (Frobenius or Spectral) norm increases with decreasing test error, whereas the average α decreases, as expected.

Consider average metrics measured on ResNet20, trained on CIFAR10, before and after apply-



(a) λ_{max} for ResNet20 layers



(b) α for ResNet20 layers

Figure 5: Analysis of ResNet20, distilled with Group Regularization, as implemented in the `distiller` (4D_regularized_5Lremoved) pre-trained models. Comparison of individual layer $\mathbf{W}_l$ maximum eigenvalues (λ_{max}, or Spectral Norms) and PL exponent α, between baseline (green) and fine-tuned (red) pre-trained models.

ing the Group Regularization technique, as implemented in the `distiller` package.[8] We analyze the available pre-trained 4D_regularized_5Lremoved baseline and fine-tuned models. Figure 5 presents the Spectral Norm (λ_{max}) and PL exponent (α) for each individual layer weight matrix $\mathbf{W}_l$.[9] The reported baseline test accuracies ($Top1 = 91.450$ and $Top5 = 99.750$) are better than the reported fine-tuned test accuracies ($Top1 = 91.020$ and $Top5 = 99.670$). Thus, the results of Figure 2 and Figure 3 might suggest that the baseline Spectral (and Frobenius) Norm should be *smaller* than those of the layers in the fine-tuned model. In both cases (Frobenius norm results not shown), we observe the opposite. On the other hand, the α values do not systematically differ between the baseline and fine-tuned models. Also (not shown), the average (unweighted) baseline α is smaller than the fine-tuned average (as would be predicted by HT-SR Theory, on which $\hat{\alpha}$ is based).

The reason for this is that the `distiller` Group Regularization technique has the unusual effect of increasing the norms of the $\mathbf{W}$ feature 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.]

5 Comparison of NLP Transformer Models

In this section, we examine the quality metrics described in Section 3 for several NLP model architectures.

apply more generally, where they differ, e.g., how the layer Spectral Norm behaves unexpectedly, while the α-based metrics can give insight into how well trained a model is.

[michael: We show that this not only works for CV, so we look at NLP. We now look in detail at the alpha metric, the spectral norm, and the new norm. We compare good and bad models, and a series of increasingly big models trained on big data sets. We show why alpha works but spectral norm is not enough

]

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

[9] We only include layer matrices or feature maps with $M \geq 50$.

Within the past two years, nearly 100 open source, pre-trained DNNs for NLP have emerged, based on the revolutionary Transformer architecture. This includes variants of BERT, Transformer-XML, GPT, etc. The Transformer architectures consist of blocks of Attention layers, containing two large, Feed Forward (Linear) weight matrics [**?**]. In contrast to the smaller pre-Activation maps arising in Cond2D layers, Attention matrices are significantly larger, and frequently under-correlated, with generally larger Heavy Tailed Power Law exponents α (when fitting the ESD). Here, we breifly look at a few popular pretrained NLP DNNs to demonstrate that the Theory of Heavy Tailed Self-Regularization also applies to NLP models and not just CV models, and to highlight how to use the theory to identify poorly trained models where the empirical norm metrics will perform poorly.

OpenAI GPT Models. We use the *WeightWatcher* tool to analyze the OpenAI GPT and GPT2 models, which gives us the opportunity to analyze the effect of both training the same model with different size data sets, and increasing sizes of both the data set and architectures. These models have generated significant media attention because of their remarkable ability to generate fake text that appears to be real and the potential misuse of this. For this reason, the original GPT model 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 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 indentify empirical indidcators for when model has in fact been poorly trained and may perform poorly when deployed.

[charles: more here ?] We analyze GPT models 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 GPT and GPT 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), resp, whereas GPT2 has input Embeddings of shape (50257×768) and (1024×768), resp. Interstingly, they also have very spectral properties, also shown below.

The OpenAI GPT2 (English) models are: [gpt-small, gpt-medium, gpt-large, and gpt-xl], having include $12, 24, 36,$ and 48 layers, resp., with increasingly larger weight matrices. The model card for GPT2 is published on github.[10]. Table 2 reports results for the average log norm metrics, using *weightwatcher (0.2.7)*, and with fully reproducible Jupyter notebooks.[11]

Empirical Quality Metrics for GPT and GPT2. We first compare the distribution of Heavy Tailed Power Law exponents α in GPT and GPT2. They are very differenmt, with GPT2 having both a notably smaller mean α, and far fewer, unusually large outliers. Figure **??** shows the empirical density (histogram) of α for all layers in GPT (blue) and GPT2 (red). [charles: discuss more]

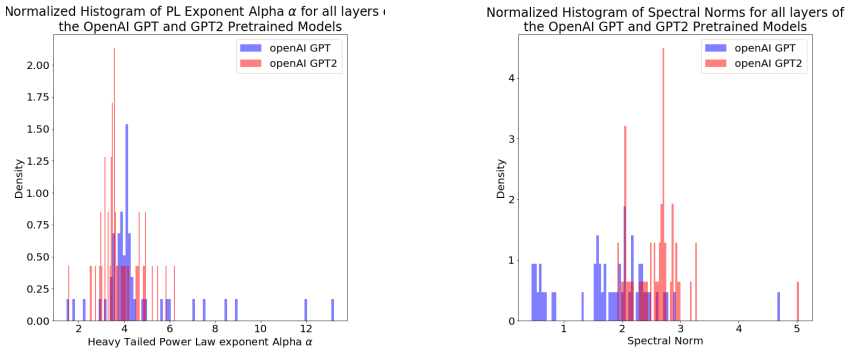
Indeed, the α metric makes a good quality metric for comparing these models. Figure 6(a) compares the PL exponent α for the GPT and GPT2 models for all layers, As expected, the improved GPT2 model has, on average, smaller α than the older GPT, with all $\alpha \leq 6$. Indeed,

[10] https://github.com/openai/gpt-2/blob/master/model_card.md

[11] https://github.com/CalculatedContent/kdd2020

Series	#Layers	Frobenius Norm $\|\mathbf{W}\|_F$	Spectral Norm $\|\mathbf{W}\|_\infty$	Weighted Alpha $\hat{\alpha} = \alpha \log \lambda_{max}$	Alpha-Norm $\|\mathbf{X}\|_\alpha^\alpha$
GPT	49	1.64	1.72	7.01	7.28
GPT (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 Log Norm Metrics for pretrainnd OpenAI GPT and GPT2 models.



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

Figure 6: Comparison of Heavy Tailed Power Law exponents α, and log Spectral Norms $\log \|\mathbf{W}\|_\infty$ for the OpenAI GPT and GPT2 (small) pretrained models.

the deficient GPT model has numerous weight matrices with unsually large fitted exponents, indicating that they are not Heavy Tailed at all. Indeed, we may expect that a poorly trained model will not exhibit consistent Heavy Tailed behavior in all layers.

However, as seen in Figure 6(b), the poorly trained GPT model has many smaller (log) Spectral Norms $\log \|\mathbf{W}\|_\infty$ than the GPT2, which would be inconsistent with a belief that smaller Spectral Norm is always better. Indeed, because there are so many anamously small $\|\mathbf{W}\|_\infty$, it appears that the GPT model may be exhibiting a kind of rank collapse. 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.

Note that Figure 6(b) also shows a couple anomalously large Spectral Norms. From Figure 7(b) (below), we see that these correspond to the first embedding layer(s). These layers appear to have a normalization, and therefore a differenc scale. For example, in GPT, most layers, the maximum eigenvalue $\lambda_{max} \sim \mathcal{O}(10 - 100)$, but in the first embedding layer, the maximum is of order N (the number of words in the embedding), or $\lambda_{max} \sim \mathcal{O}(10^5)$. For GPT and GPT2, we layer 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). 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 6(b).

Correlation Flow in GPT and GPT2. also differs significantly between GPT and GPT2. Figure 7(a) plots α vs the depth (i.e. a layer id) for each model.

(a) Power Law Exponent (α)

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

(c) log Alpha-Norm ($\log \|\mathbf{X}\|_\alpha^\alpha$)

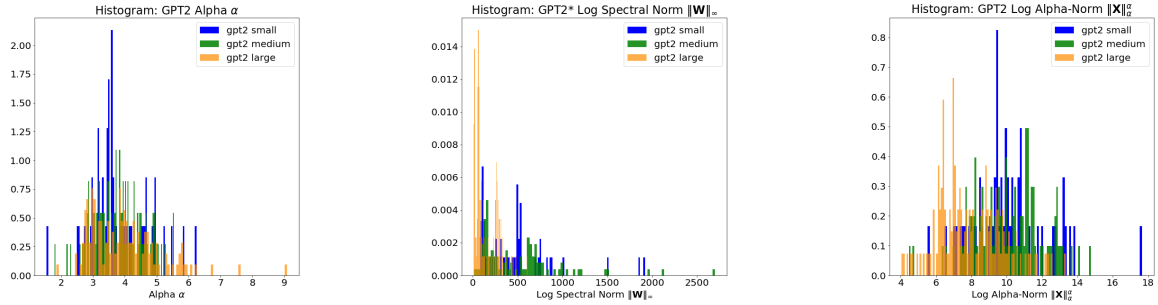
Figure 7: Comparison of Correlation Flow and Spectral Norm for OpenAI GPT and GPT2

[charles: Discuss Spectral Norm, alpha-Norm]

GPT2: small, medium, large, xl We now look across the series of increasingly improving GPT2 models, examining both our PL exponent α, as well as the various Norm metrics. As we move from small to xl, both the PL exponents α, and the peak of distribtion of log Norm metrics shifts from larger to smaller values. Figure 8 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.

We see that the average α decreases with increasing model size, althogh, the differences are less noticible between GTP2 mkodels than between the GPT and GPT2 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 8(b) shows decreasing average log Spectral Norms with the larger models. As we have seen in the trends of other 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 overparameterized and/or not yet fully optimized and require datasets even larger than the recent XL $1.5B$ release [**?**].



(a) Power Law Exponent (α)

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

(c) log Alpha-Norm ($\log \|\mathbf{X}\|_\alpha^\alpha$)

Figure 8: Comparison of Power Law Exponnents, Spectral Norm, and Alpha-Norm for different size models in the GPT2 architecture series. *Note: the log spectral norms (b) histogram omits the first 2 layers, corresponding to the embedding layer, which are normalized differently and have anamolously large values.

6 Comparison of all pretrained 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 466 pretrained computer vision models Pytorch. These image classification and segmentation models are pretrained on nine datasets, ImageNet-1K, and CIFAR-10, CIFAR-100, Street 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: We actually don't run regressions on all these datasetsm, BUT we could present them in the Figure below to show that alpha is a good metric for these kinds of models, in contrast to the NLP, where alpha is frequently too large to run a regression]

[charles: insert link in the footnote]

and a full summary of all the models analyzed is included in the Appendix. For our analysis, we then group models by architecture and datasets for further analysis.

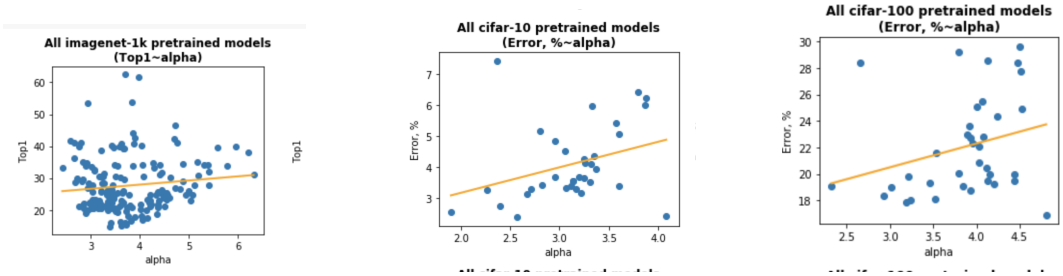
In this paper, we propose that the Weightwatcher tool could be used to predict the trends in the generalization accuracy of deep neutral network without a test set. To test our proposition, we choose simple linear regression to analyze the relationship between the Weightwatcher metrics and the traditional accuracy metric obtained with a test set (we avoid polynomial regressions as they are more prone to overfitting and does not make economic sense). On the left-hand-side of regression, we have the Top1 errors, Top5 errors as reported for ImageNet-1K models, Error

[charles: Question: only the alpha-related metrics is the core metrics of the WW tool, right? Metrics such as stable rank or spectral norm have been suggested in other paper before???]

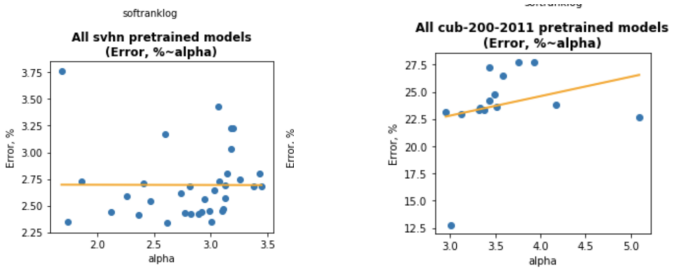
[charles: insert a simple formula for linear regression]

To further refine 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

[charles: see examples in Figure X]



(a) ImageNet 1K (b) CIFAR 10 (c) CIFAR 100



(d) SVHN (e) CUB 200

Figure 9: [charles: Preliminary charts:] Heavy Tailed Power Law exponent α vs. reported Top1 Test Accuracies for pretrained DNNs available[charles: ref] for 5 different data sets.

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

Table 3: Datasets used

(Here we omit the results for CUB-200-2011, Pascal-VOC2012, ADE20K, and COCO datasets as there are less than 15 models for those datasets and thus the regression is less statistically significant)

[charles: insert plots for Figure X]

For the second batch, we plot the regression for models of each architecture-datasets combination, which shows the relationship between the progression of the model accuracy and Weightwatcher metrics more clearly and precisely. For example, as you could see in the Figure X2,

[charles: Add an example, UPDATE when we have the results from the new codes]

Insert plots for Figure X2

While running each regression, we record the R-squared and mean squared errors (MSE) for each regression. We then filter out regressions with less than five datapoints and models with structural outliers.

[charles: Define and give an example of the structural outliers]

[charles: What are the structrual outliers we chose ?]

Insert a plot for outliers, OPTIONAL

[charles: These tables could go to appendix. We need references]

[charles: Still preliminary, only 309 data points here]

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 4: Architectures used

	Frobenius Norm $\langle \log \|\mathbf{W}\|_F \rangle$	Spectral Norm $\langle \log \|\mathbf{W}\|_\infty \rangle$	Weighted Alpha $\langle \hat{\alpha} = \alpha \log \lambda_{max} \rangle$	Alpha-Norm $\langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle$
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 5: 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 [charles: and which are postively correlated with the test error?]. These results can be readily reproduced using the Google Colab notebooks accompanying thie paper [**?**]

Results

7 Conclusion

XXX. PUT CONCLUSION HERE AND WEAVE IN COMMENTS FROM BELOW.

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 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.
- 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.

Acknowledgements. MWM would like to acknowledge ARO, DARPA, NSF, and ONR for providing partial support of this work.

A Appendix

XXX. APPENDIX.