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

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

Abstract

XXX. ABSTRACT.

GLG — Powered by Box

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 *pre-trained model*—and be told to use it.

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 theory to guide that practice. Moreover, if ML is to become an industrial process, then the 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 the loss of values of the hyperparameters were, 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 pre-trained 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 apply techniques to evaluate the generalization properties of large-scale state-of-the-art pre-trained neural network (NN) models.[1] To do so, we consider a large

*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.

[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 pre-trained NN models.

suite of publicly-available models 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 pre-trained 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] XXX. MORE DETAILS. Importantly, for all of these models, we have no access to training data or testing data.

XXX. LIST PLACES WHERE THEY ARE AVAIALBLE, HERE OR IN NEXT SECTION.

In more detail, our main contributions are the following.

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

2 Background and Related Work

Here we will cite and discuss related work, including

[?], [?], [?], [?] [?], [?], [?],

3 Methods

We assume we are given several pretrained Deep Neural Networks (DNNs), as part of a similar architecture. We would like to estimate the trends in the reported test / generalization accuracy accross a series of similar archtectures. and without Batch Normalization, trained on ImageNet, and widely available in the pyTorch distribution.

Empirical Complexity Metrics To do this, we will compute a variety of *Complexity Metrics* based on the Product Norm of the layer weight matrics. Note that unlike traditional ML approaches, however, we do not seek a bound on the complexity (i.e. test error), nor are we trying to evaluating a single model with differing hyperparmeters. We wish to examine different models a common architecture series. And, also, compare different architectures themselves.

Let us write the Energy Landscape (or optimization function) for a typical DNN with L layers as

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

with activation functions $h_l(\cdot)$, weight matrices $\mathbf{W}_l$, and the biases $\mathbf{b}_l$.

The model has been (or will be) trained on (unspecified) labeled data $\{d_i, y_i\} \in \mathcal{D}$, using Backprop, by minimizing some (also unspecified) loss function $\mathcal{L}()$. Moreover, we expect that most well trained,. production quality models will employ 1 or more forms of on regularization, such as Batch Normalization, Dropout, etc, and will also contain additional structure such as Skip Connections etc. Here, we ignore these details, and focus only on the weight matrices.

Each layer contains by one or more layer 2D weight matrices $\mathbf{W}_L$, and/or the 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.) We assume the layer weight matrices are statistically independent,

[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.

allowing us to estimate the Complexity $\mathcal{C}$, or test accuracy, with a standard Product Norm, which resembles a data dependent VC complexity

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

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:

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

Here, we will consider the following Norms:

- Frobenius Norm: $\|\mathbf{W}\|_F^2 = \|\mathbf{W}\|_2^2 = \sum_{i,j} w_{i,j}^2$
- Spectral Norm: $\|\mathbf{W}\|_\infty = \lambda_{max}$
- $\alpha-$Norm (or Shatten Norm) $\|\mathbf{X}\|_\alpha^\alpha = \sum_{i=1}^M \lambda^\alpha$,

where λ_i is the $i - th$ eigenvalue of the *Empirical Correlation Matrix*

$$\mathbf{X} = \mathbf{W}^T \mathbf{W} \tag{3}$$

and λ_{max} is the maximum eigenvalue. These eigenvalues are the square of the singular values σ_i of $\mathbf{W} : \lambda_i = \sigma_i^2$.

The exponent α is the power law exponent that arises in our *Theory of Heavy Tailed Self Regularization*, and is the determined by fitting the Empirical Spectral Density (ESD) of $\mathbf{X}$–i.e. a histogram of the eigenvalues–$\rho(\lambda)$ to a truncated power law

$$\rho(\lambda) \sim \lambda^\alpha, \;\; \lambda \leq \lambda_{max} \tag{4}$$

We will also consider an approximate capacity metric, $\hat{\alpha}$, shown previously to correlate well with the trends in reported test accuracies of pretrained DNNs [**?**]

- $\hat{\alpha} = \alpha \log \lambda_{max} \approx \log \|\mathbf{X}\|_\alpha^\alpha$

which approximates the log $\alpha-$Norm for both Very Heavy Tailed weight matrices ($\alpha < 2$) and reasoably well for finite size, Moderately Heavy Tailed ones $\alpha \in [2, 5]$.

Spectral Analysis of Convolutional 2D Layers While we can easily analyze Linear layers, there is some ambiguity in performing spectral analysis on Convolutional 2D (Conv2D) layers. A Conv2D layer is 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 and output channels, respectively. Typically, the $w = h = k$, giving $(k \times k)$ tensor slices, or *pre-Activation Maps* $\mathbf{W}_{i,L}$ of dimension $(in\_ch \times out\_ch)$ each. Usually $in\_ch \leq out\_ch$.

There are at least 3 different approaches to computing the Singular Values Decomposition(s) (SVD) of an Conv2D layer

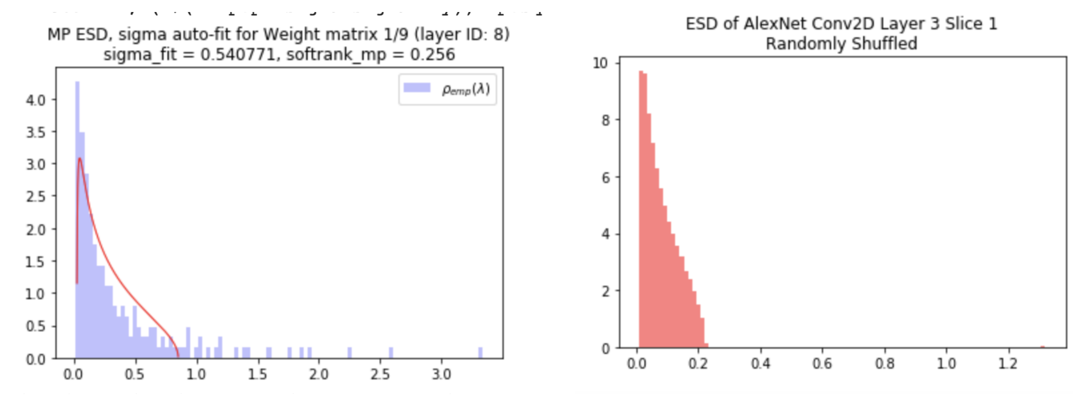
1. run SVD on each of the pre-Activation Maps $\mathbf{W}_{i,L}$, yielding $(k \times k)$ sets of M singular values.
2. stack the feature maps into a single rectangular matrix of, say, dimension $((k \times k \times out\_ch) \times in\_ch)$, yielding in_ch singular values

3. Compute the 2D Fourier Transform (FFT) for each of the (in_ch, out_ch) pairs, and run SVD on the resulting Fourier coeffients[?]. This leads to $\sim (k \times in\_ch \times out\_ch)$ non-zero singular values.

Each method has tradeoffs. While, in principle, 3. is mathematically sound, it is computationally expensive. For this study, because we are computing tens of thousands of calculations, we select 1., which is numerically the fastest and easiest to reproduce.[4]

To verify that our approach is meaningful, we need to see 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 Conv2D 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.

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.



(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

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.

Normalization of Empirical Matrices . The normalization does not affect the Power Law fits because the Heavy Tailed exponent α is scale-invariant. As are metrics like the Stable Rank and our MP Soft Rank. In this study, however, all of the metrics strongly depend on the scale of the weight matrix. Formally, to apply RMT, we would usually define Correlation Matrix with

[4] We will provide a Google Colab notebook where all results can be reproduced, with the option to redo the calculations with option 3 for the SVD of the Conv2D.

$\frac{1}{N}$ normalization, and assume that either the variance of $\mathbf{X}$ is unity, $\sigma^2 = 1$, or is known. [5] But for these empirical studies, the pretrained DNNs are typically initialized with random weight matrices $\mathbf{W}_0$, with the variance already normalized to $\sqrt{\frac{1}{N}}$, or some variant of this such as Glorot (or Xavier) Normalization[?], or as $\sqrt{\frac{2}{N*k^2}}$ for Convolutional 2D Layers. We do not normalize (or renormalize) the Empirical Correlation Matrices, and use them as-is, except that we rescale the Conv2D pre-Activation Maps $\mathbf{W}_{i,L}$ by $\frac{k}{\sqrt{2}}$ so that they are on the same scale as the Linear layers.

Note that in some pretrained models, like the pyTorch VGG models, the Linear weight matrices appear to have been initialized with variance $\sigma \sim 0.01$. We do not attempt to correct for this, and, instead, simply try to treat all models the same.

COMMENT ON HOW LOG NORM first and last layers behave, maybe somewhere else

COMMENT ON HOW LOG PORM for GPT includes unusually high alpha, not meaningful other than to show the trend.

The WeightWatcher Tool

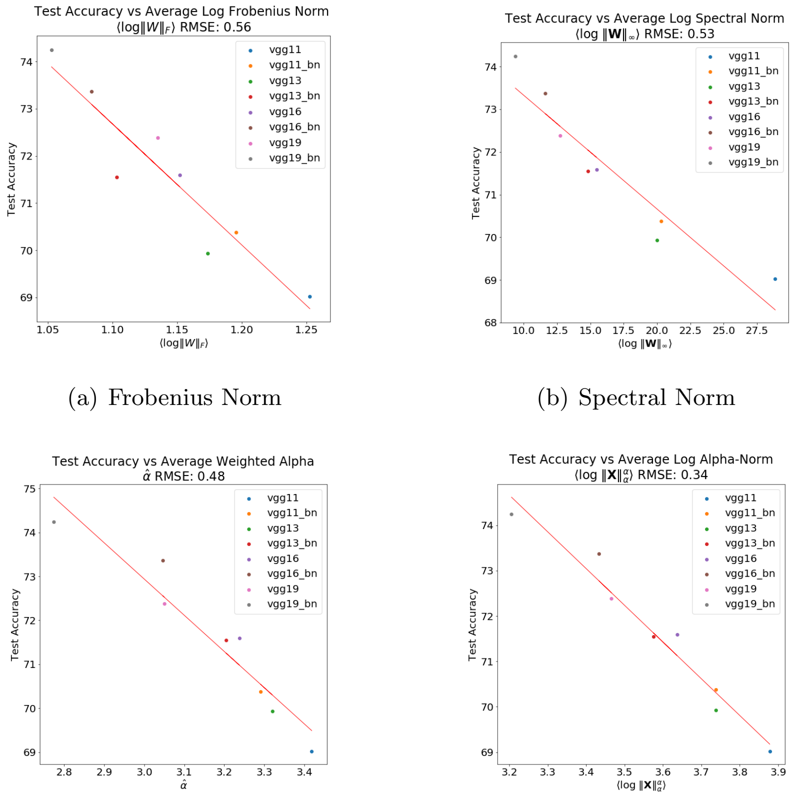
Organization of this paper [charles: This could be moved to the end of the into also] We want to study the effectiveness of product norm metrics in predicting trends in the generalization accuracy of real-word, production quality, pretrained Deep Neural Networks. In Section 4, we study three well known, and widely available DNN CV architectures: the VGG, the ResNet, and the DenseNet series of models. We lay out the basic methodology we use to evaluate the different metrics against the reported test accuracies. We also introduce the notion of Correlation Flow, and use this to compare the three pretrained architectures in detail. Then, in Section 5, we look at several variations of a popular NLP DNN architecture, the OpenAI GPT and GPT2 models. We discuss when and how the different metrics apply more generally, and where they differ. In particular, we show where the layer Spectral Norm $\|\mathbf{W}\|_\infty$ behaves unexpectedly, and, contrastly, how the α parameter can give insight into how well trained a model is. Finally, in Section **??**, we examine over 100 pretraind DNN CV models, and show how well each metric is correlated with the reported test accuracies across different models in an architeture series, and argue the Alpha-Norm metric(s) perform remarkably well. —-

4 Comparison of 3 Computer Vision Models

We start our empirical studies by compute the differernt log product norm metrics for three computer vision (CV) architectures, the VGG, and ResNet, and the DenseNet series. Each series consists of several pretrained DNN models, trained on the full ImageNet[?] dataset, and distributed with the current opensource pyTorch framework (vesion 1.4)[?]. We also provide 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"[?]; we call this the ResNet-1K series of models. We compute the metrics using the WeightWatcher tool (version 0.2.7), and provide Jupyter notebooks in the github repo for this paper[?], making the results fully reproducible.

Figure 2 plots the different log norm metrics versus the reported Test accuracies[?], for the pretrained models: VGG11, VGG13, VGG16, and VGG19, with and without BatchNormalization. The figure also shows a basic linear regression line, with the Root Mean Squared Error

[5] And for formal proofs of the Heavy Tailed results, we need a different normalization, $\frac{1}{N^{1-\alpha}}$



(a) Frobenius Norm

(b) Spectral Norm

(c) Weighted Alpha

(d) Alpha-Norm

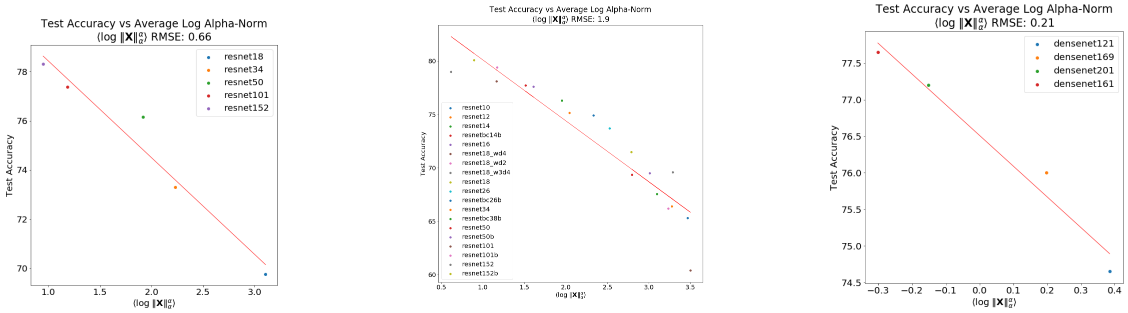
Figure 2: Comparison of norm metrics vs reported test accuracy for pretrained VGG models, trained on ImageNet, available in pyTorch. Plots will be updated and replace

(RMSE) shown. All four metrics correlate well with the reported Top1 Accuracies, with smaller norms implying better generalziation (greater accuracy, lower error). Similar results are obtained for the Top5 Accuracies (not shown). Notice that the log α-Norm metric $\log\|\mathbf{W}\|)_\infty^\infty$ performs best, with an RMSE of 0.34. Also, the next best metric is the Weighted Alpha, $\hat{\alpha}$, which is an approximate log α-Norm metric.

Empirical Metrics vs Test Accuracies for CV models Table 1 summarizes results from linear regressions, applied to the VGG, ResNet, ResNet-1K, and DenseNet series, and reports the RMSE for all four product norm metrics. For all series, all four metrics correlates remarkably well with the reported Top1 test accuracies. Moreover, the α-Norm and/or the approximate Weighted Alpha metrics perform best.

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.

Correlation Flow in CV Models We can learn even more about how a pretrained model will perform by examining what we call the *Correlation Flow* down the layers. By this, we mean the values of individual Heavy Tailed Power Law exponent α for each layer weight matric $\mathbf{W}$, as



(a) ResNet

(b) ResNet-1K

(c) DenseNet

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

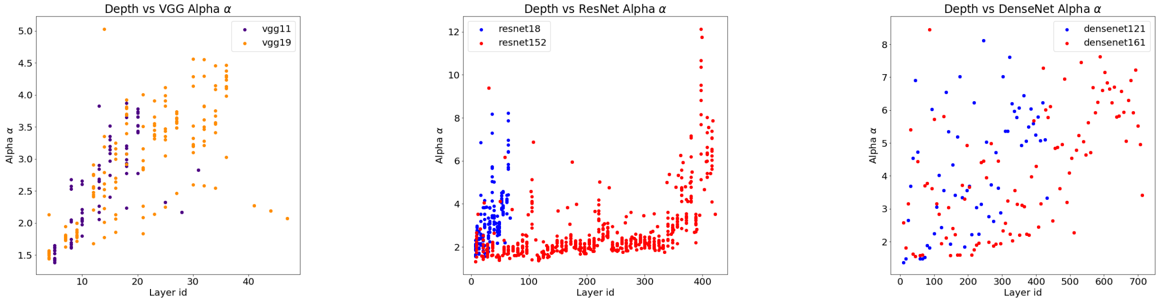
Series	#Models	Frobenius Norm $\|\mathbf{W}\|_F$	Spectral Norm $\|\mathbf{W}\|_\infty$	Weighted Alpha $\hat{\alpha} = \alpha \log \lambda_{max}$	Alpha-Norm $\|\mathbf{X}\|_\alpha^\alpha$
VGG	6	0.56	0.53	0.48	0.42
ResNet	5	0.9	1.4	0.61	0.66
ResNet-1K	19	2.4	3.6	1.8	1.9
DenseNet	4	0.3	0.26	0.16	0.21

Table 1: RMSE for Linear Fits of Metric to Reported Top1 Test Error for all pre-trained models in the architecture series. All models trained on ImageNet except ResNet-1K, which was trained on ImageNet-1K.

a function of depth (or layer id) in the network. (The first layer corresponds to the data, and the last layer the labels). Figure 4 plots α for each layer for the VGG (no batchnorm), RseNet-1K, and DenseNet models, including the least accurate and most accurate model in each series.

First observe that in the VGG models, α systematically increases as we move down the network from data to labels, in the Conv2D layers, starting with $\alpha \leq 2.0$ and reaching all the way to $\alpha \sim 5.0$. Then, $\alpha \in [2, 2.5]$ stabilizes back down to last three, large, fully connected (FC) layers. In contrast, in the larger ResNet-1K (152) model, α remaims stable at $\alpha \sim 2.0$ all the way until we reach the final layers. Finally, in the DenseNet models, as in the VGG models, α increases as the layer id increases, although it displays a large variance, and can range all the way to $alpha \leq 8.0$.

[charles: INTERPRET]



(a) VGG

(b) ResNet-1K

(c) DenseNet

Figure 4: Correlation Flow: Heavy Tailed Power Law exponent α vs layer for VGG, ResNet-1K, and DenseNet models.

5 Comparison of NLP Transformer Models

[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

]

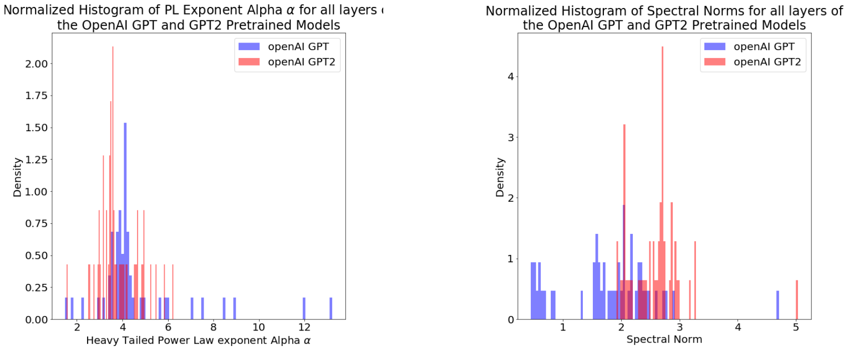
In the past two years, nearly 100 open source, pre-trained DNNs for Natural Language Processing (NLP) have emerged, based on the revolutionary Transformer architecture, including variants of BERT, Transformer-XML, GPT, etc. The Transformer architectures consist of blocks of Attention layers, containing of 2 large, Feed Forward (Linear) weight matrics [**?**]. In contrast to the smaller pre-Activation maps arising in Cond2D layers, the 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

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 5: 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.

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.

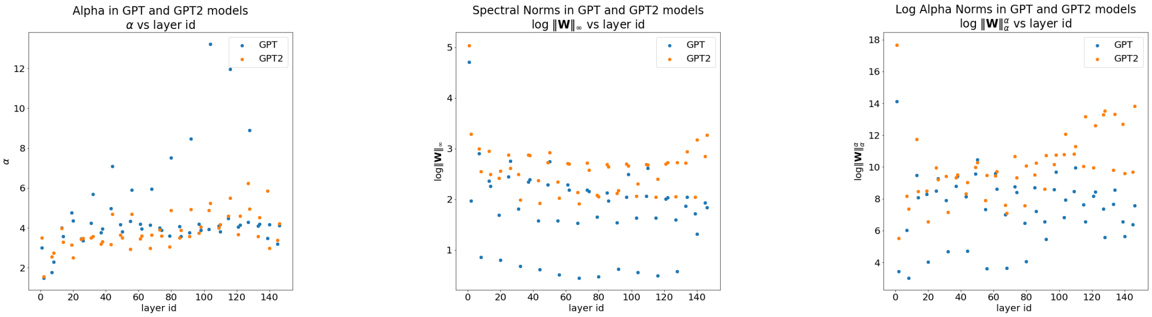
The OpenAI GPT2 (English) models are: [gpt-small, gpt-medium, gpt-large, and gpt-xl], having include $12, 24, 36,$ and 48 layers, resp., with increasingly larger weight matrices. The model card for GPT2 is published on github.[6]. Table 2 reports results for the average log norm metrics, using *weightwatcher (0.2.7)*, and with fully reproducible Jupyter notebooks.[7]

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 5(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, the deficient GPT model has numerous weight matrices with unsually large fitted exponents,

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

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



(a) Power Law Exponent (α)

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

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

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

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 5(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.

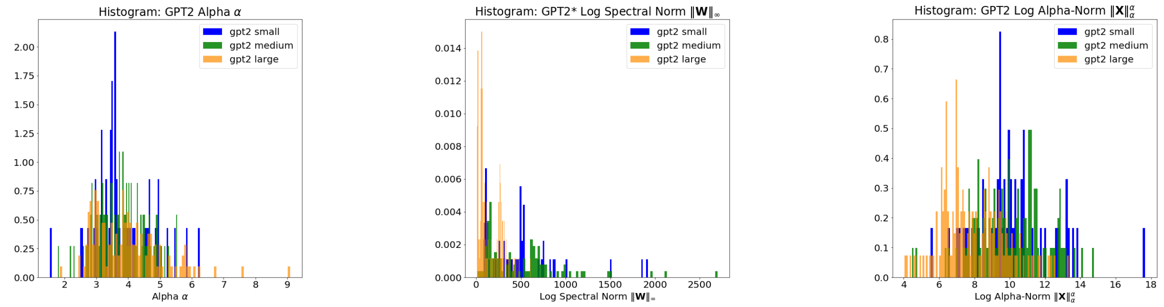
Note that Figure 5(b) also shows a couple anomalously large Spectral Norms. From Figure 6(b) (below), we see that these correspond to the first embedding layer(s). These layers appear to have a normalization, and therefore a differenc scale. For example, in GPT, most layers, the maximum eigenvalue $\lambda_{max}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

The Correlation Flow in GPT and GPT2 also differs significantly between GPT and GPT2. Figure 6(a) plots α vs the depth (i.e. a layer id) for each model.

[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 7 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 7(b) shows decreasing average log Spectral Norms with the larger models. As we have



(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 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.

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[**?**].

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

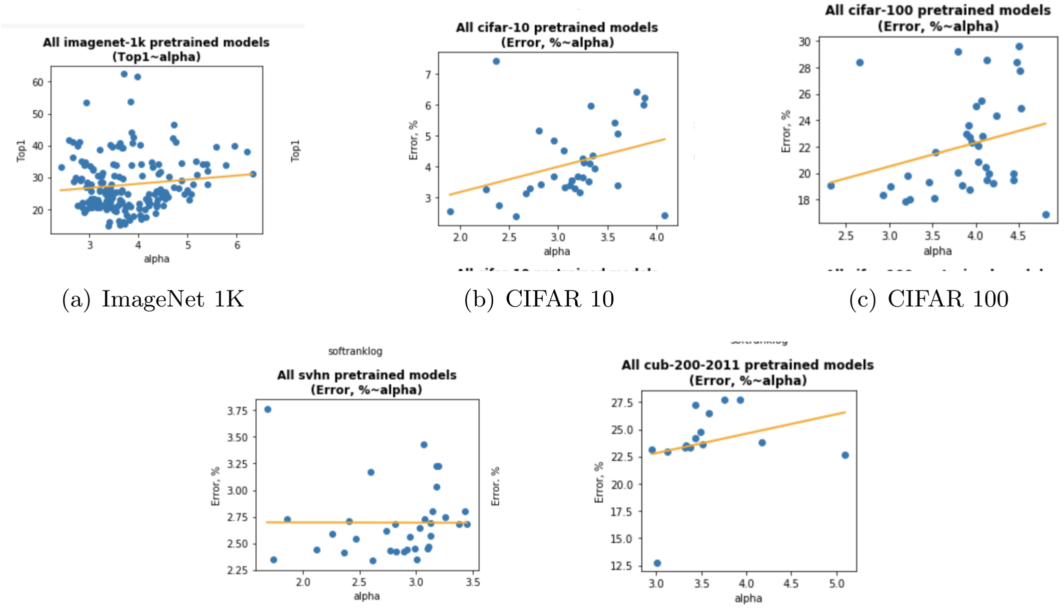
[charles: We actually don't run regressions on all these datasetsm, BUT we could present them in the Figure below to show that alpha is a good metric for these kinds of models, in contrast to the NLP, where alpha is frequently too large to run a regression]

[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???]



(a) ImageNet 1K

(b) CIFAR 10

(c) CIFAR 100

(d) SVHN

(e) CUB 200

Figure 8: [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.

[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]

(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]

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

Table 3: 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 4: Architectures used

Results

	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[**?**]

7 Conclusion

XXX. CONCLUSION.

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