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. For example, below we compare the 8 pretrained models in the VGG series: (VGG11, VGG11_BN$\cdots$ VGG19), with and without Batch Normalization, trained on ImageNet, and widely available in the pyTorch distribution.

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 correlation matrix $\mathbf{X} = \frac{1}{N}\mathbf{W}^T\mathbf{W}$, and λ_{max} is the maximum eigenvalue. (Note that 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

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

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 $\alpha \in [2, 5]$.

SOME MORE DISCUSSION HERE

[charles: not actually feature Maps, need to find the correct term: Tensor Slice]

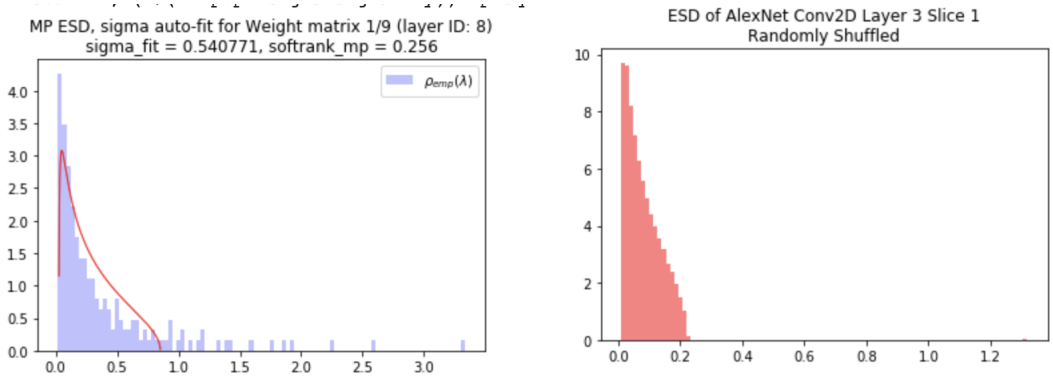
Spectral Analysis of Convolutional 2D Layers While we can easily analyze Linear layers, there is some ambiguity in performing spectral analysis on Convolutional 2D (Conv2D) layers. A Conv2D layer is a 4-index tensor of dimension (w, h, in_ch, out_ch), specified by an $(w \times h)$ filter, and in_ch, out_ch input and output channels, respectively. Typically, the $w = h = k$, giving $(k \times k)$ Tensor Slices $\mathbf{W}_{i,L}$ of dimension $(in\_ch \times out\_ch)$ each. Usually $in\_ch \ll out\_ch$. 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 Tensor Slices $\mathbf{W}_{i,L}$, 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) ESD of AlexNet Conv2D Layer 3 Slice 1 (b) ESD of the same matrix, randomly shuffled

Figure 1: Caption

These plots tell us that the Tensor Slice 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 metric.

—-

[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.

4 Comparison of 3 Computer Vision Models

5 Comparison of NLP Transformer Models

6 Comparison of Metrics on PreTrained CV Models

7 Conclusion

XXX. CONCLUSION.

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