Practical Theory and Neural Network Models

Michael W. Mahoney

ICSI and Dept of Statistics, UC Berkeley

http://www.stat.berkeley.edu/∼mmahoney/

April 2021

(Joint work with Charles H. Martin,
Calculation Consulting, charles@calculationconsulting.com)

Outline

A motivating question

Given a SOTA CV or NLP model, can we (or how can we, e.g., what metric, with or without any data) tell if it is overparameterized?

A motivating question

~~Given a SOTA CV or NLP model, can we (or how can we, e.g., what metric, with or without any data) tell if it is overparameterized?~~

Can we predict trends in the quality of state-of-the-art neural networks without access to training or testing data?*

* "Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data," Martin, Peng, and Mahoney, arXiv:2002.06716, Accepted for publication, *Nature Communications*.

A motivating question

~~Given a SOTA CV or NLP model, can we (or how can we, e.g., what metric, with or without any data) tell if it is overparameterized?~~

Can we predict trends in the quality of state-of-the-art neural networks without access to training or testing data?*

- Odd question for AI/ML people – if forced, they say of course not.
- Some other possible answers:
  - Yes or no, since a theorem says such-and-such.
  - Yes or no, if you assume some Bayesian something-or-other.
  - Yes or no, if distributional covariates do such-and-such.
  - Maybe, since convolutions smooth, but not for NLP.
  - I don't know, since I build systems that work for any data.
- This is *not* how people build bridges or do brain surgery or explore for oil or trade stocks or ...
- Why is it the way we do AI/ML?

* "Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data," Martin, Peng, and Mahoney, arXiv:2002.06716, Accepted for publication, *Nature Communications*.

What is theory? What is the role of theory?

https://en.wikipedia.org/wiki/Theory

Scientific theory:

- "a well-confirmed type of explanation of nature, made in a way consistent with scientific method . . . described in such a way that scientific tests should be able to provide empirical support for it, or empirical contradiction ("falsify") of it."
- descriptive: this is the way the world is

Mathematical theory:

- "a branch of or topic in mathematics . . . an extensive, structured collection of theorems"
- prescriptive/normative: this is the way the world should be

"Working with state-of-the-art neural network models is a practical business, and it demands a practical theory."

This matters . . . and guides the type of theory we prefer

Describing versus prescribing:

- Use a new metric to make a falsifiable prediction.
- Use a new metric to make a regularizer for training.

Shape parameters versus size parameters:

- Size/norm of matrix = norm of eigenvalues.
- Shape of eigenvalue distribution = more refined information.

Constraints on the type of theory:

- Worst-case bounds work better with "norms."
- Shape of eigenvalues related to "volumes."

Determining causes from data.

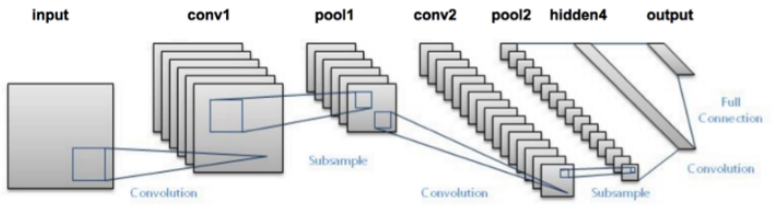
- Does a bounding theorem or a metrics on the data establish causality?
- Does successfully making falsifiable predictions establish causality?

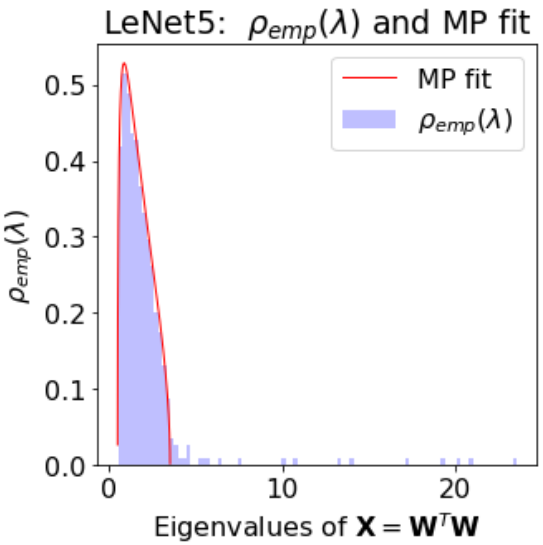
Outline

Lots of DNNs analyzed: Look at nearly every publicly-available SOTA model in CV and NLP

- *Don't evaluate your method on one/two/three NNs, evaluate it on:*
  - *dozens (2017)*
  - *hundreds (2019)*
  - *thousands (2021)*
- *Don't use bad/toy models, use SOTA models.*
  - *If you do, don't be surprised if low-quality/toy models are different than high-quality/SOTA models.*
- *Don't train models, instead validate pre-trained models.*
  - *Validating models is harder than training models.*

Results: LeNet5 (an old/small NN example)





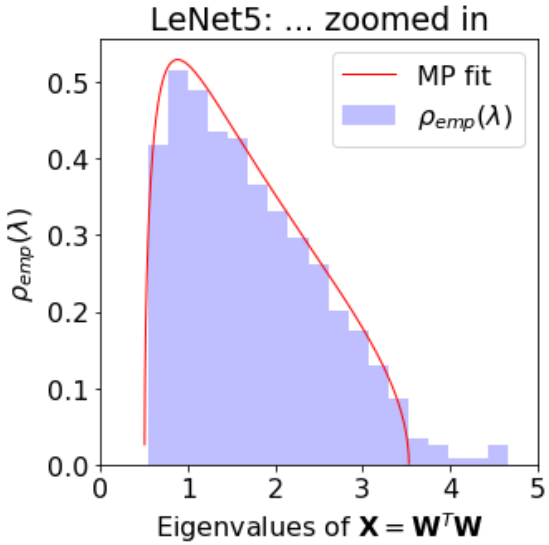
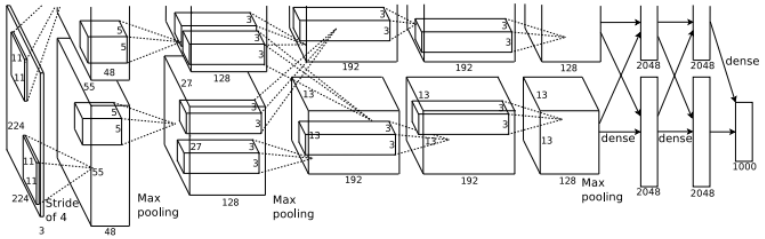
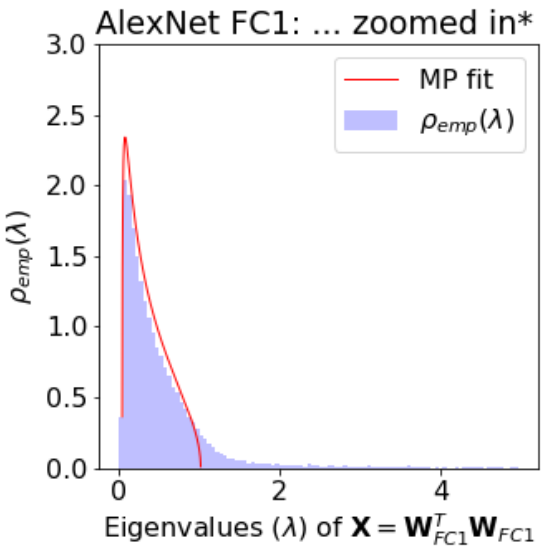


Figure: Full and zoomed-in ESD for LeNet5, Layer FC1.

Older and/or smaller and/or less well-trained models look like bulk+spike.

Results: AlexNet (a typical modern/large DNN example)





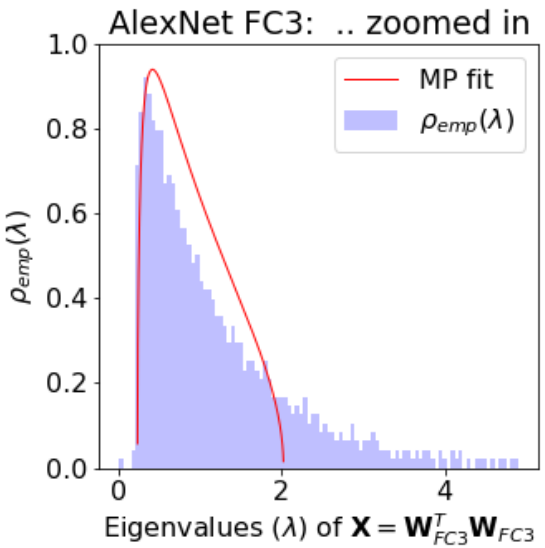
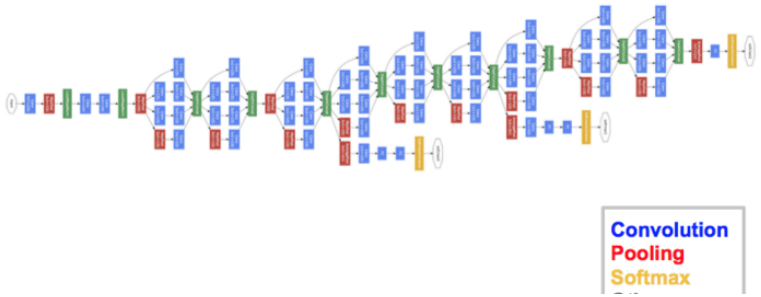
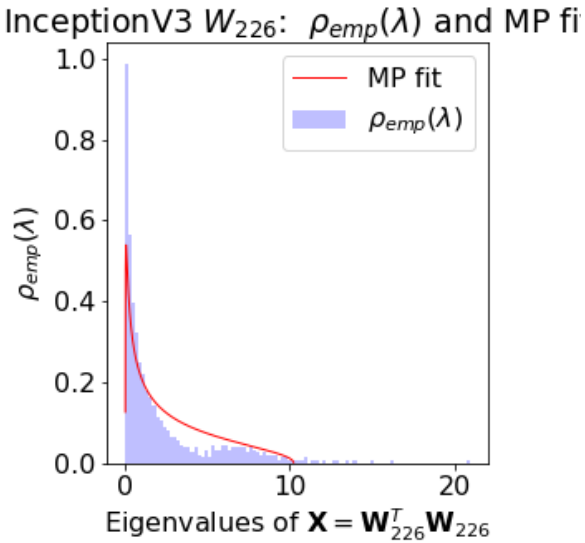


Figure: Zoomed-in ESD for Layer FC1 and FC3 of AlexNet.

Newer SOTA models have heavy-tail structure in their weight matrix correlations (i.e., not elements but eigenvalues).

Results: InceptionV3 (one particularly unusual example)





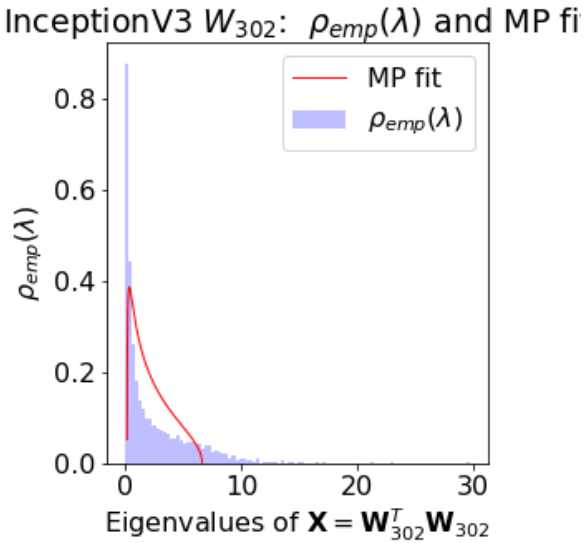
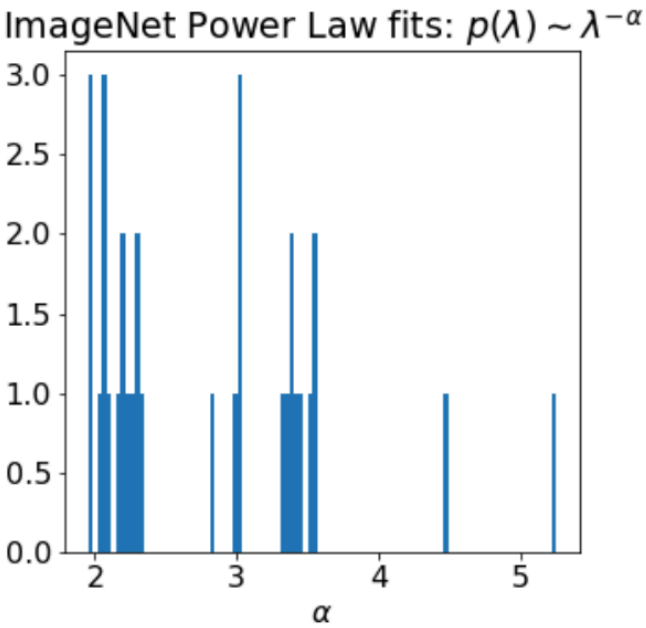


Figure: ESD for Layers L226 and L302 in InceptionV3, as distributed w/ pyTorch.

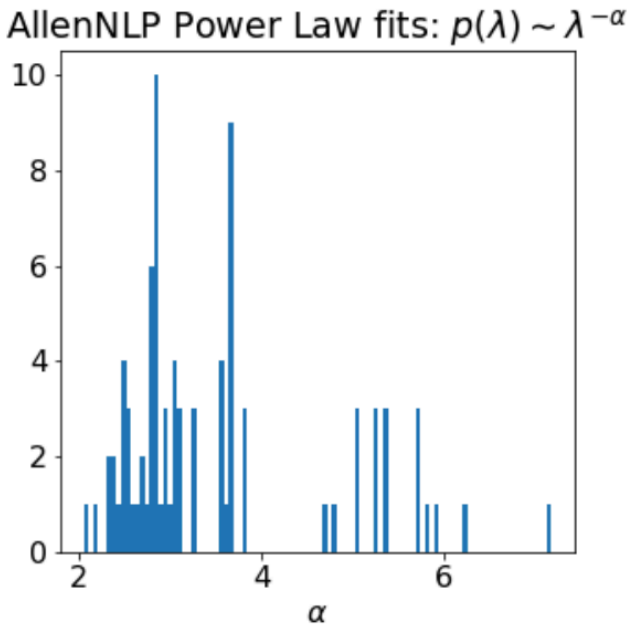
Lots of "funny stuff" in real data—many models are exceptions, some are "exceptions that prove the rule."

Ubiquity of heavy-tailed ESDs: ImageNet and AllenNLP

(older results, from ca. 2018.)



(a) ImageNet pyTorch models



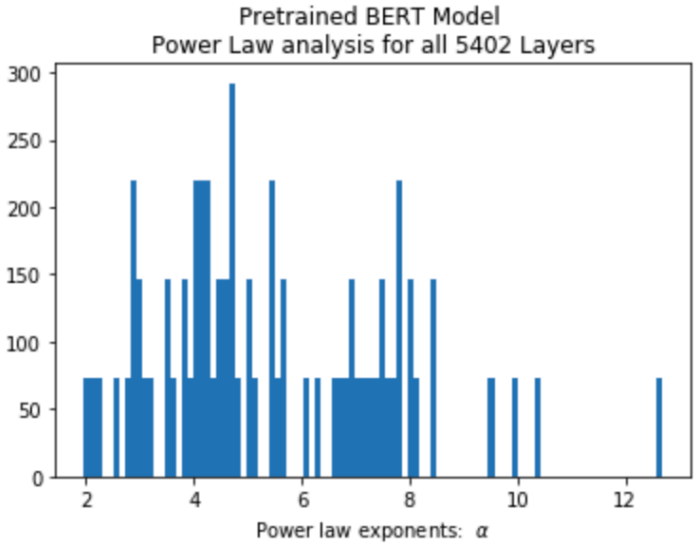
(b) AllenNLP models

Figure 12: Distribution of power law exponents α for linear layers in pre-trained models trained on ImageNet, available in pyTorch, and for those NLP models, available in AllenNLP.

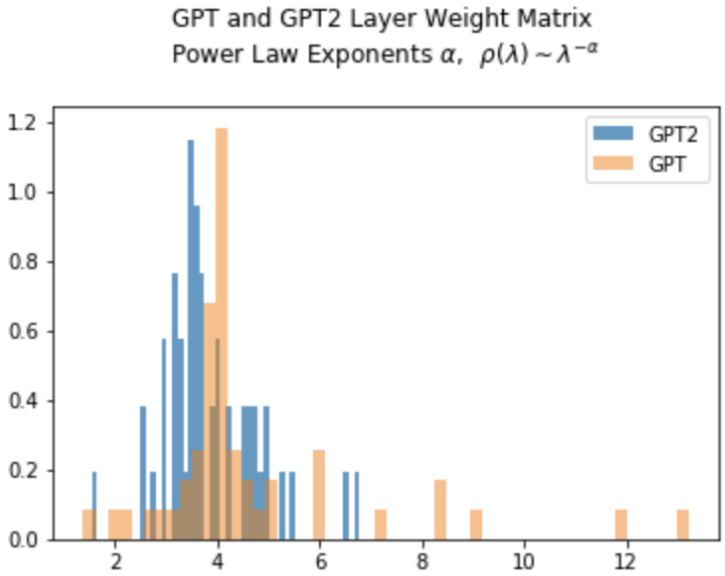
All these models display remarkable Heavy Tailed Universality

Ubiquity of heavy-tailed ESDs: BERT and GPT vs GPT2

(older results, from ca. 2018.)



(a) The pretrained BERT model is *not* optimal (has large exponents and displays rank collapse)

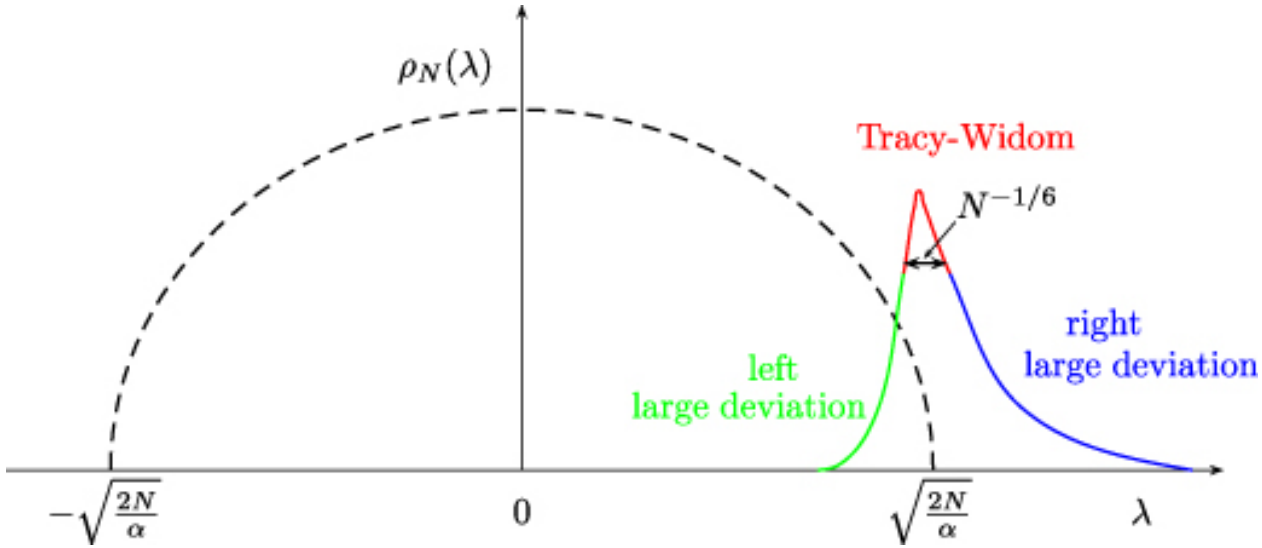


(b) GPT versus GPT2: example of a class of models that "improves" over time.

Outline

Random Matrix Theory 101: Wigner and Tracy-Widom

- Wigner: *global bulk statistics* approach universal semi-circular form
- Tracy-Widom: *local edge statistics* fluctuate in universal way



Problems with Wigner and Tracy-Widom:

- Weight matrices usually not square
- Typically do only a single training run

Random Matrix Theory 102: Marchenko-Pastur

Let $\mathbf{W}$ be an $N \times M$ random matrix, with elements $W_{ij} \sim N(0, \sigma^2_{mp})$.

Then, the ESD of $\mathbf{X} = \mathbf{W}^T\mathbf{W}$, converges to a deterministic function:

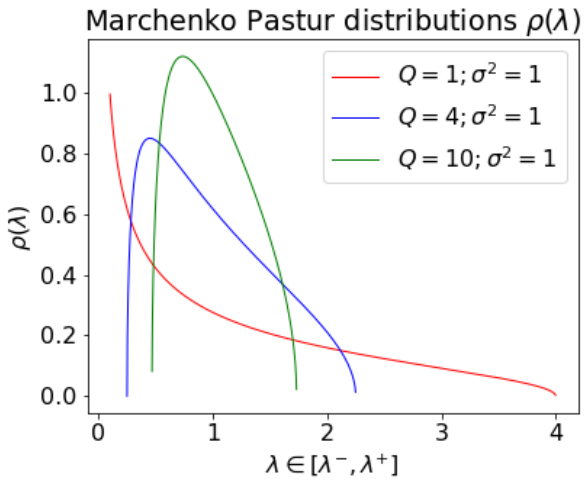
$$\rho_N(\lambda) := \frac{1}{N}\sum_{i=1}^{M}\delta(\lambda - \lambda_i)$$

$$\xrightarrow[Q \text{ fixed}]{N\to\infty} \begin{cases} \frac{Q}{2\pi\sigma^2_{mp}}\frac{\sqrt{(\lambda^+ - \lambda)(\lambda - \lambda^-)}}{\lambda} & \text{if } \lambda \in [\lambda^-, \lambda^+] \\ 0 & \text{otherwise.} \end{cases}$$

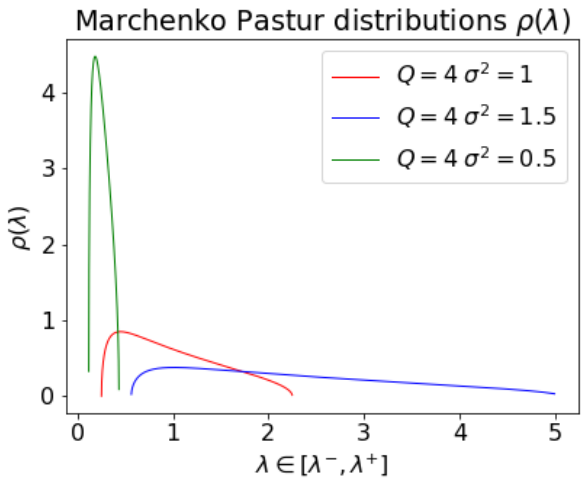
with well-defined edges (which depend on Q, the aspect ratio):

$$\lambda^{\pm} = \sigma^2_{mp}\left(1 \pm \frac{1}{\sqrt{Q}}\right)^2 \qquad Q = N/M \geq 1.$$

Random Matrix Theory 102': Marchenko-Pastur



(c) Vary aspect ratios



(d) Vary variance parameters

Figure: Marchenko-Pastur (MP) distributions.

Important points:

- *Global bulk stats*: The overall shape is deterministic, fixed by Q and σ.
- *Local edge stats*: The edge λ^+ is very crisp, i.e., $\Delta\lambda_M = |\lambda_{max} - \lambda^+| \sim O(M^{-2/3})$, plus Tracy-Widom fluctuations.

We use both *global bulk statistics* as well as *local edge statistics* in our theory.

Random Matrix Theory 103: Heavy-tailed RMT

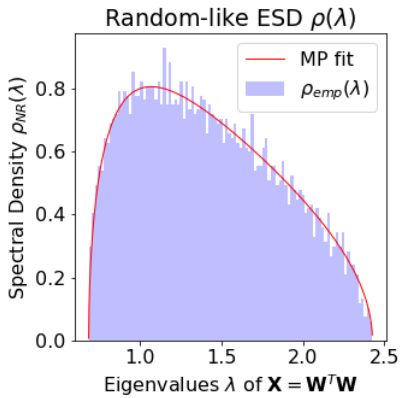
Go beyond the (relatively easy) Gaussian Universality class:

- *model* strongly-correlated systems ("signal") with heavy-tailed random matrices.

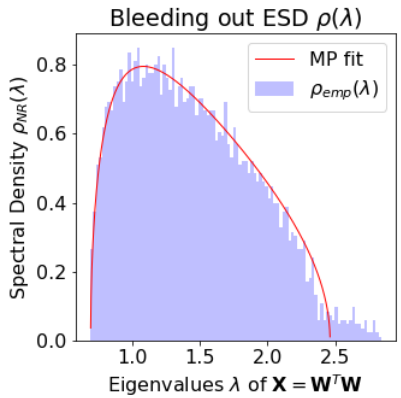
	Generative Model w/ elements from Universality class	Finite-N Global shape $\rho_N(\lambda)$	Limiting Global shape $\rho(\lambda),\ N \to \infty$	Bulk edge Local stats $\lambda \approx \lambda^+$	(far) Tail Local stats $\lambda \approx \lambda_{max}$
Basic MP	Gaussian	MP distribution	MP	TW	No tail.
Spiked-Covariance	Gaussian, + low-rank perturbations	MP + Gaussian spikes	MP	TW	Gaussian
Heavy tail, $4 < \mu$	(Weakly) Heavy-Tailed	MP + PL tail	MP	Heavy-Tailed*	Heavy-Tailed*
Heavy tail, $2 < \mu < 4$	(Moderately) Heavy-Tailed (or "fat tailed")	PL** $\sim \lambda^{-(a\mu+b)}$	PL $\sim \lambda^{-(\frac{1}{2}\mu+1)}$	No edge.	Frechet
Heavy tail, $0 < \mu < 2$	(Very) Heavy-Tailed	PL** $\sim \lambda^{-(\frac{1}{2}\mu+1)}$	PL $\sim \lambda^{-(\frac{1}{2}\mu+1)}$	No edge.	Frechet

Basic MP theory, and the spiked and Heavy-Tailed extensions we use, including known, empirically-observed, and conjectured relations between them. Boxes marked "*" are best described as following "TW with large finite size corrections" that are likely Heavy-Tailed, leading to bulk edge statistics and far tail statistics that are indistinguishable. Boxes marked "**" are phenomenological fits, describing large ($2 < \mu < 4$) or small ($0 < \mu < 2$) finite-size corrections on $N \to \infty$ behavior.

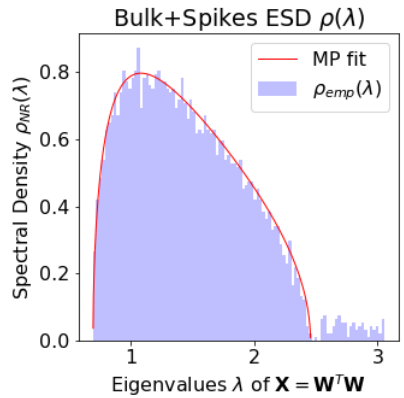
RMT-based 5+1 Phases of Training (in pictures)



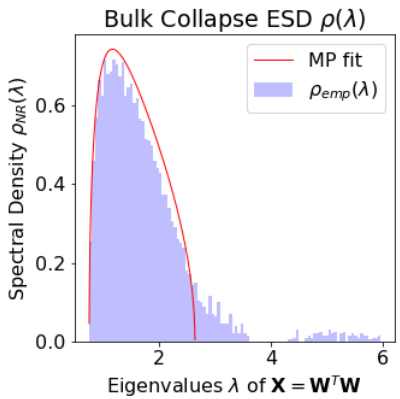
(a) RANDOM-LIKE.



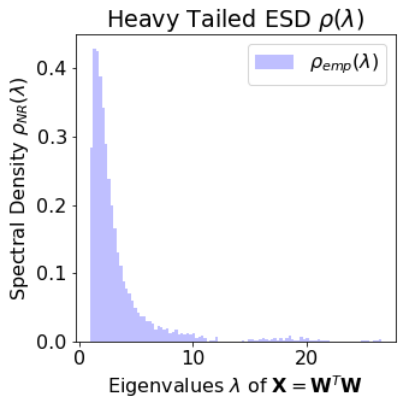
(b) BLEEDING-OUT.



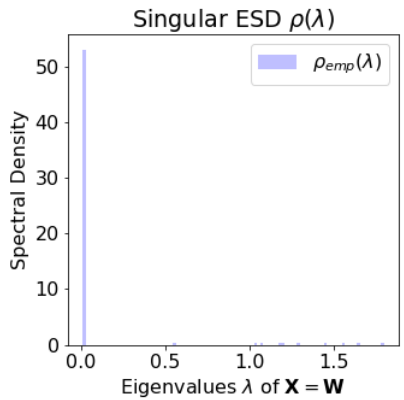
(c) BULK+SPIKES.



(d) BULK-DECAY.



(e) HEAVY-TAILED.



(f) RANK-COLLAPSE.

Figure: The 5+1 phases of learning we identified in DNN training.

Bulk+Spikes: Small Models ∼ Tikhonov regularization

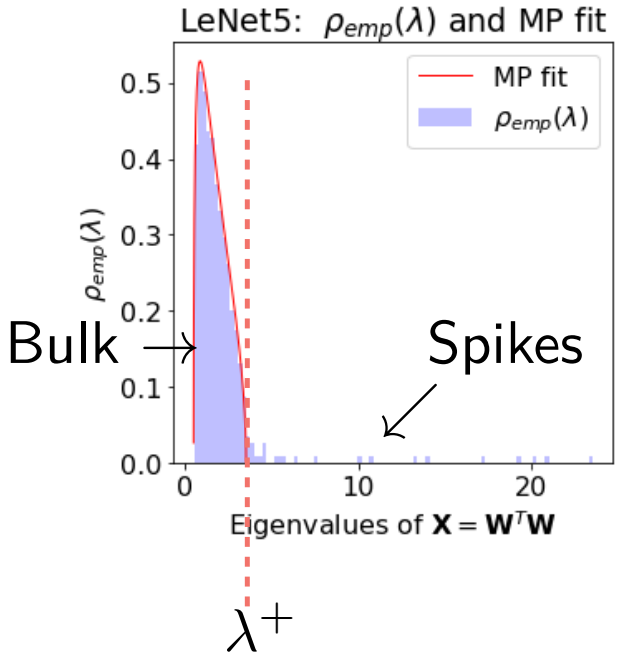
Low-rank perturbation

$$\mathbf{W}_l \simeq \mathbf{W}_l^{rand} + \Delta^{large}$$

Perturbative correction

$$\lambda_{max} = \sigma^2 \left(\frac{1}{Q} + \frac{|\Delta|^2}{N}\right)\left(1 + \frac{N}{|\Delta|^2}\right)$$

$$|\Delta| > (Q)^{-\frac{1}{4}}$$



simple scale threshold

$$\mathbf{x} = \left(\hat{\mathbf{X}} + \alpha\mathbf{I}\right)^{-1} \hat{\mathbf{W}}^T\mathbf{y}$$

eigenvalues $> \alpha$ (Spikes) carry most of the signal/information

Smaller, older models like LeNet5 exhibit traditional regularization and can be described perturbatively with Gaussian RMT

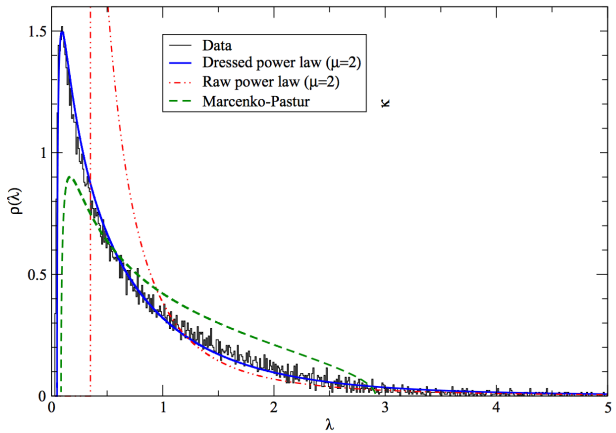
Heavy-tailed Self-regularization

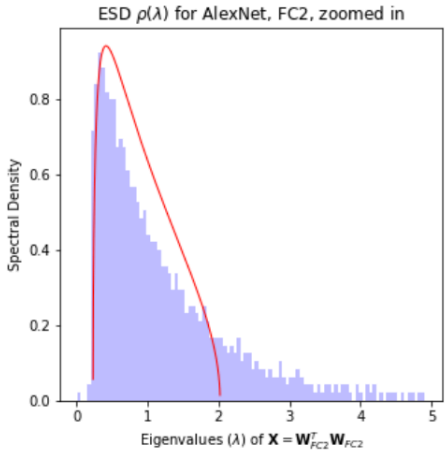
W is *strongly-correlated* and highly non-random

- We *model* strongly-correlated systems by heavy-tailed random matrices
- We *model* signal (not noise) by heavy-tailed random matrices

Then RMT/MP ESD will also have heavy tails.

- The eigenvalues are heavy-tailed; the weights are NOT.





"All" larger, modern DNNs exhibit novel Heavy-tailed self-regularization

Mechanisms and regularization

Mechanisms:

- Multiple mechanisms can give rise to heavy-tailed ESDs
- We do *not* need to posit/understand a mechanism to use the theory

Every adjustable *knob* and *switch** is regularization:

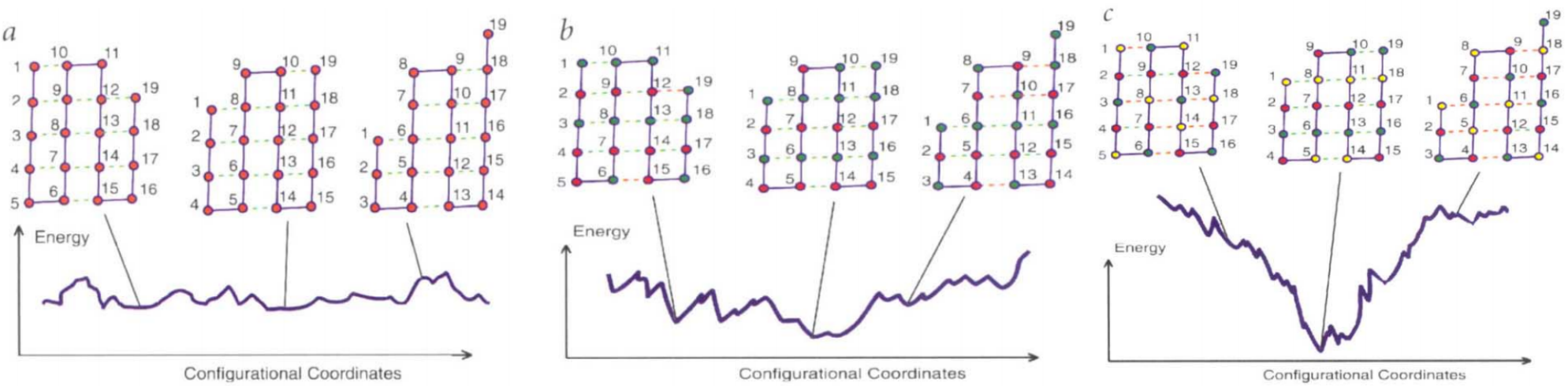
- "Explicit regularization": replace $\min f$ with $\min f + \lambda g$
- "Implicit regularization" = regularization due to non-approximate computation†
- "Heavy-tailed self regularization" = regularization due to the data: SOTA models train to good data, not bad or random or arbitrary data

*and there are *many* (https://arxiv.org/pdf/1710.10686.pdf)

†see "Approximate Computation and Implicit Regularization ...", Mahoney, arXiv:1203.0786, PODS12

Implications: Minimizing Frustration and Energy Funnels

As simple as can be?, Wolynes, 1997



Energy Landscape Theory: "random heteropolymer" versus "natural protein" folding

Somewhat like Moore-Penrose inverse for least-squares . . .

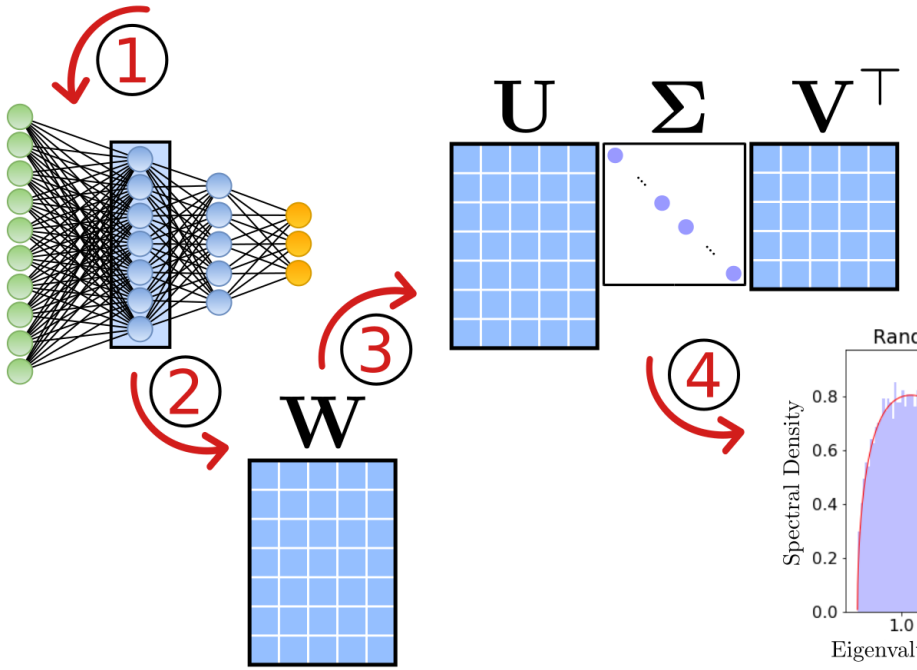
- Every regime: gives minimum norm solution
- Underdetermined regime: many nearby solutions (since it is singular perturbation)
- Overdetermined regime: few nearby solutions (depending on smoothness parameters)

. . . except here the model depends on properties of the data.

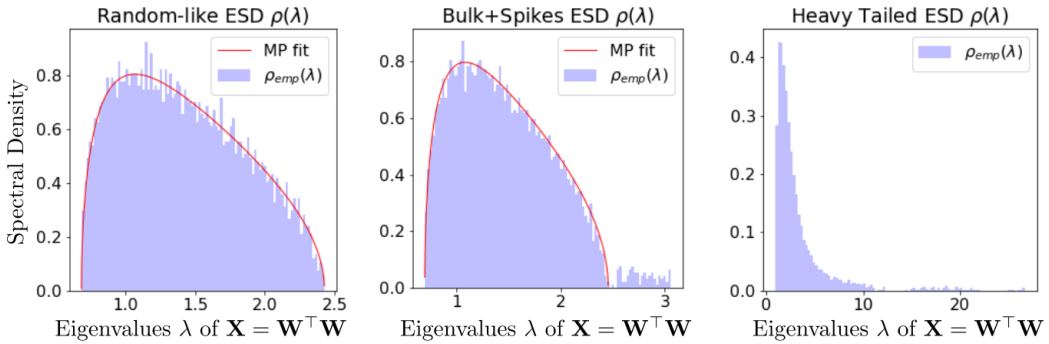
Watching weights with WeightWatcher

https://github.com/CalculatedContent/WeightWatcher

Analyzing DNN Weight matrices with WeightWatcher



1. Take a model
   - 2. Take a weight matrix
     - 3. Do Spectral analysis
       - 4. Histogram of eigenvalues



- Analyze one layer of pre-trained model
- Compare multiple layers of pre-trained model
- Monitor NN properties as you train your own model

"pip install weightwatcher"

Outline

Using the theory

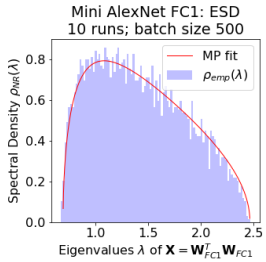
Different ways one could *use* a theory.

- Perform diagnostics for model validation, to develop hypotheses, etc.*
- Make predictions about model quality, generalization, transferability, etc.*
- Did post-training modifications damage my model?*
- Will buying more data help?*
- Will training longer help?*
- Will quantizing or distilling help?*
- Construct a regularizer to do model training.**

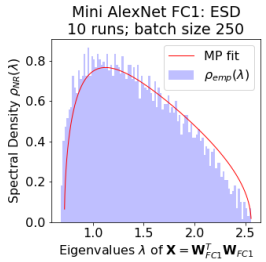
*Ideally, by peeking at very little or no data.

**If you have lots of data, lots of GPUs, etc.

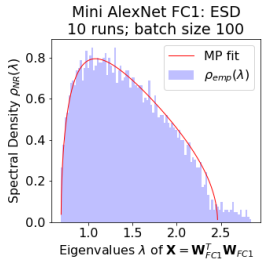
Batch Size Tuning: Exhibiting the Phases



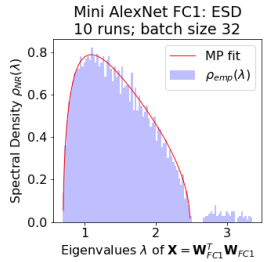
(a) Batch Size 500.



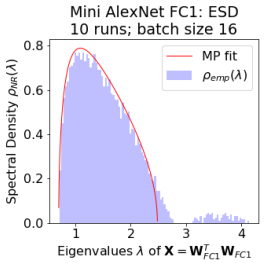
(b) Batch Size 250.



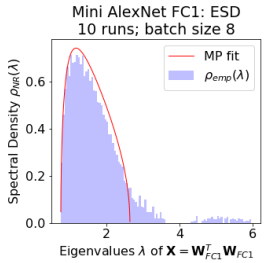
(c) Batch Size 100.



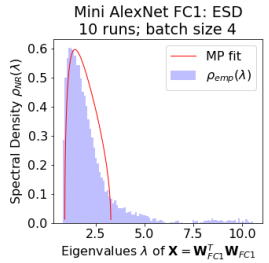
(d) Batch Size 32.



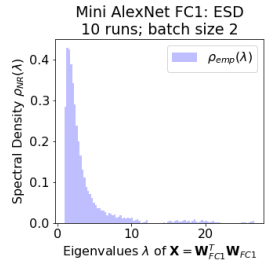
(e) Batch Size 16.



(f) Batch Size 8.



(g) Batch Size 4.



(h) Batch Size 2.

Figure: Varying Batch Size. ESD for Layer FC1 of MiniAlexNet. We exhibit all 5 of the main phases of training by varying only the batch size.

- *Decreasing batch size induces strong correlations in* $\mathbf{W}$*, leading to a more implicitly-regularized model.*
- *Increasing batch size washes out strong correlations in* $\mathbf{W}$*, leading to a less implicitly-regularized model.*

Predicting test accuracies ... lots of metrics ...

- Average log norm (a VC-like data-dependent capacity metric):

$$\langle \log \|\mathbf{W}\| \rangle = \frac{1}{N} \sum_{l,i} \log \|\mathbf{W}_{l,i}\| = \frac{1}{N} \sum_{l,i} \log(\lambda_{l,i}^{max})$$

- Average alpha (also data-dependent, from HT-SR theory):

$$\alpha = \frac{1}{N} \sum_{l,i} \alpha_{l,i}$$

- Combine the two into a weighted average (weighted to compensate for different size and scale of feature maps):

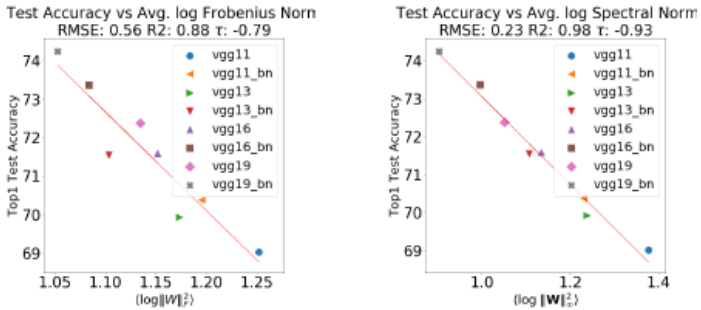
$$\hat{\alpha} = \frac{1}{N} \sum_{l,i} \log(\lambda_{l,i}^{max}) \alpha_{l,i}$$

- In a special case ($\alpha \approx 2$), for each layer:

PL–Norm Relation: $\alpha \log \lambda^{max} \approx \log \|\mathbf{W}\|_F^2$.

"pip install weightwatcher"

(The first) large-scale study (meta-analysis) of hundreds of SOTA pretrained models [‡]



(a) Log Frobenius Norm, VGG (b) Log Spectral Norm, VGG

(c) Weighted Alpha, VGG (d) Log α-Norm, VGG

Figure 2: Comparison of Average Log Norm and Weighted Alpha quality metrics versus re test accuracy for pretrained VGG models: VGG11, VGG13, VGG16, and VGG19, with and

Different metrics on pre-trained VGG.

Series	#	Metric	$\langle \log \|\mathbf{W}\|_F^2 \rangle$	$\langle \log \|\mathbf{W}\|_\infty^2 \rangle$	$\hat{\alpha}$	$\langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle$
VGG	6	RMSE	0.56	**0.23**	0.48	0.34
		R^2	0.88	**0.98**	0.92	0.96
		Kendall-τ	-0.79	**-0.93**	**-0.93**	**-0.93**
ResNet	5	RMSE	0.9	0.97	**0.61**	0.66
		R^2	0.92	0.9	**0.96**	0.9
		Kendall-τ	-1.0	-1.0	-1.0	-1.0
ResNet-1K	19	RMSE	2.4	2.8	**1.8**	1.9
		R^2	0.81	0.74	**0.89**	0.88
		Kendall-τ	-0.79	-0.79	**-0.89**	-0.88
DenseNet	4	RMSE	0.3	**0.11**	0.16	0.21
		R^2	0.93	**0.99**	0.98	0.97
		Kendall-τ	-1.0	-1.0	-1.0	-1.0

Table 1: Quality metrics (for RMSE, smaller is better; for R^2, larger is better; and for Kendall-τ rank correlation, larger magnitude is better) for reported Top1 test error for pretrained models in each architecture series. Column # refers to number of models. VGG, ResNet, and DenseNet were pretrained on ImageNet. ResNet-1K was pretrained on ImageNet-1K.

Summary statistics: VGG; ResNet; DenseNet.

	$\log \|\cdot\|_F^2$	$\log \|\cdot\|_\infty^2$	$\hat{\alpha}$	$\log \|\cdot\|_\alpha^\alpha$
RMSE (mean)	4.84	5.57	4.58	4.55
RMSE (std)	9.14	9.16	9.16	9.17
R2 (mean)	3.9	3.85	3.89	3.89
R2 (std)	9.34	9.36	9.34	9.34
Kendal-tau (mean)	3.84	3.77	3.86	3.85
Kendal-tau (std)	9.37	9.4	9.36	9.36

Table 3: Comparison of linear regression fits for different average Log Norm and Weighted Alpha metrics across 5 CV datasets, 17 architectures, covering 108 (out of over 400) different pretrained

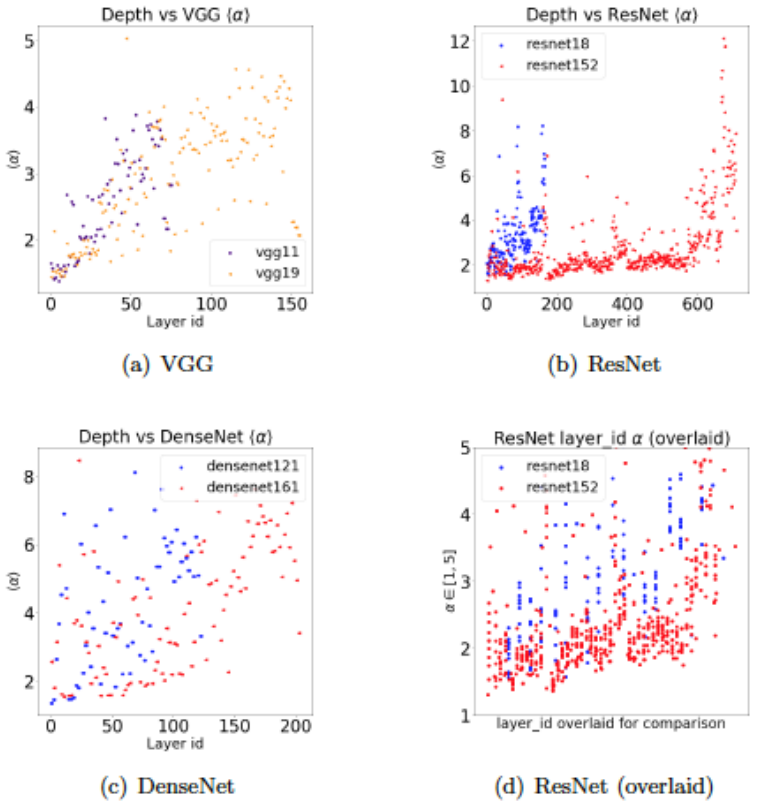
Summary statistics: hundreds of models.

Lots more plots to prove we can "predict trends . . . without access . . . "

[‡] "Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data," Martin, Peng, and Mahoney, arXiv:2002.06716, Accepted for publication, *Nature Communications*.

Using a theory: on SOTA models

Analyzing pre-trained models.



(a) VGG (b) ResNet

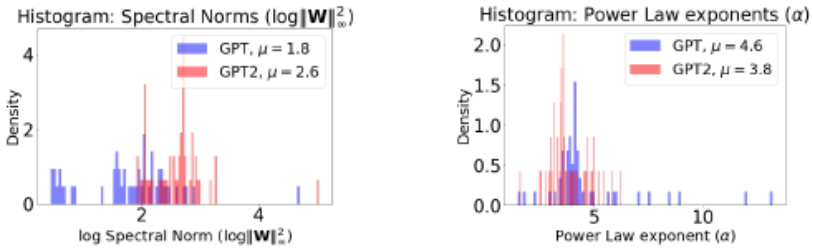
(c) DenseNet (d) ResNet (overlaid)

Figure 4: PL exponent (α) versus layer id, for the least and the most accurate models in VGG (a), ResNet (b), and DenseNet (c) series. (VGG is without BN; and note that the Y axes on

Alpha versus depth: VGG, ResNet, DenseNet.

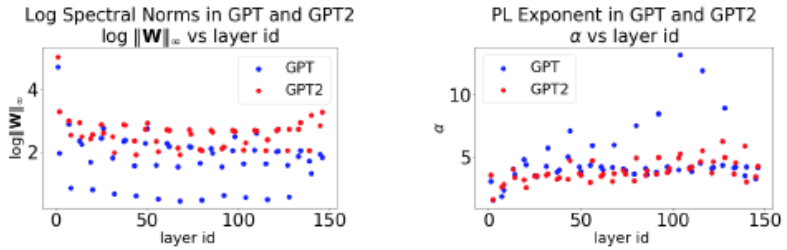
Using a theory: on SOTA models

Analyzing pre-trained models.



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

Figure 6: Histogram of PL exponents and Log Spectral Norms for weight matrices from the OpenAI GPT and GPT2-small pretrained models.



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

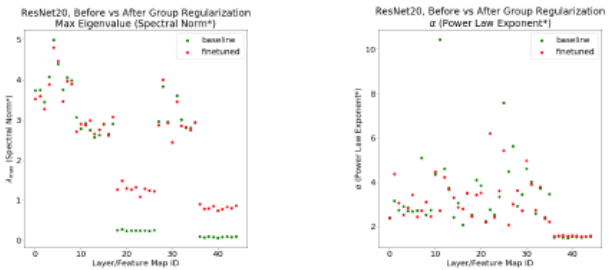
Figure 7: Log Spectral Norms (in (a)) and PL exponents (in (b)) for weight matrices from the OpenAI GPT and GPT2-small pretrained models. (Note that the quantities shown on each Y axis are different.) In the text, this is interpreted in terms of Scale Collapse and Correlation Flow.

Histogram and depth plots of $\alpha_{l,i}$ and $\lambda_{l,i}^{max}$.

Using a theory: easy to break popular SLT metrics

Easy to "break" popular SLT metrics

- they are *not* validated counterfactually
- they drive the development of models



(a) λ_{max} for ResNet20 layers (b) α for ResNet20 layers

Figure 5: ResNet20, distilled with Group Regularization, as implemented in the `distiller` (4D_regularized_5Lremoved) pretrained models. Log Spectral Norm ($\log \lambda_{max}$) and PL exponent (α) for individual layers, versus layer id, for both baseline (before distillation, green) and fine-tuned (after distillation, red) pretrained models.

Series	#	$\langle \log \|\mathbf{W}\|_F \rangle$	$\langle \log \|\mathbf{W}\|_\infty \rangle$	$\hat{\alpha}$	$\langle \log \|\mathbf{X}\|_\alpha^\alpha \rangle$
GPT	49	1.64	1.72	7.01	7.28
GPT2-small	49	2.04	2.54	9.62	9.87
GPT2-medium	98	2.08	2.58	9.74	10.01
GPT2-large	146	1.85	1.99	7.67	7.94
GPT2-xl	194	1.86	1.92	7.17	7.51

Table 2: Average value for the average Log Norm and Weighted Alpha metrics for pretrained OpenAI GPT and GPT2 models. Column # refers to number of layers treated. Averages do

Intel's distillation "broke" their models.

GPTx series: how does a model trained to "bad" data differ from one trained to "good" data?

Using a theory: leads to predictions

Based on analyzing hundreds of pre-trained SOTA models:

- "**Correlation flow**":
  - "Shape" of ESD of adjacent layers, as well as overlap between eigenvectors of adjecent layers, should be well-aligned.
- "**Scale collapse**":
  - "Size" of ESD of one or more layers changes dramatically, while the size of other layers changes very little, as a function of some perturbation of a model, during training (or post-training modification).
- "**Correlation traps**":
  - Spuriously large eigenvalues[§] may appear, and they may even be important for model convergence.

We can measure these quantities with Weightwatcher—so can you!

[§] Eigenvalues not due to signal in the data—we have theorems-style theory for Hessians ("Hessian Eigenspectra of More Realistic Nonlinear Models," Liao and Mahoney, `https://arxiv.org/abs/2103.01519`), but it's still open for Weights.

More publicly-available data

A contest (Predicting Generalization in Deep Learning, NeurIPS 2020).
Our experiences:

- based on a "fantastic" paper (considered *many* metrics, but not α or $\hat{\alpha}$)
- nominally about *causes* of generalization; but, like most ML contests,
  - ensemblization—good way to win
  - information leakage—hard to avoid
  - augment data—good way to win
  - (But none of those tell us about generalization.)
- big difference between 0 error and ≈ 0 error
- not worth competing in*
- thanks to organizers for releasing data*

*since we want to understand causes of good model performance

Models and metrics

Models and tasks: can segment models by architecture parameters or solver parameters.

Series	#	(L)	Batch Sizes	Dropout	Weight Decay	Conv Widths
`Task1` "task1_v4"	0xx	4	8, 32, 512	0.0, 0.5	0.0, 0.001	256, 512
(VGG-like)	1xx	5	8, 32, 512	0.0, 0.5	0.0, 0.001	256, 512
	2xx	5	8, 32, 512	0.0, 0.5	0.0, 0.001	256, 512
	5xx	8	8, 32, 512	0.0, 0.5	0.0, 0.001	256, 512
	6xx	8	8, 32, 512	0.0, 0.5	0.0, 0.001	256, 512
	7xx	9	8, 32, 512	0.0, 0.5	0.0, 0.001	256, 512
`Task2` "task2_v1"	2xx	13	32, 512, 1024	0.0, 0.25, 0.5	0.0, 0.001	512
(Network-in-network)	6xx	7	32, 512, 1024	0.0, 0.25, 0.5	0.0, 0.001	512
	9xx	10	32, 512, 1024	0.0, 0.25, 0.5	0.0, 0.001	512
	10xx	10	32, 512, 1024	0.0, 0.25, 0.5	0.0, 0.001	512

Table 1: Overview of models (from [5, 6]) we considered.

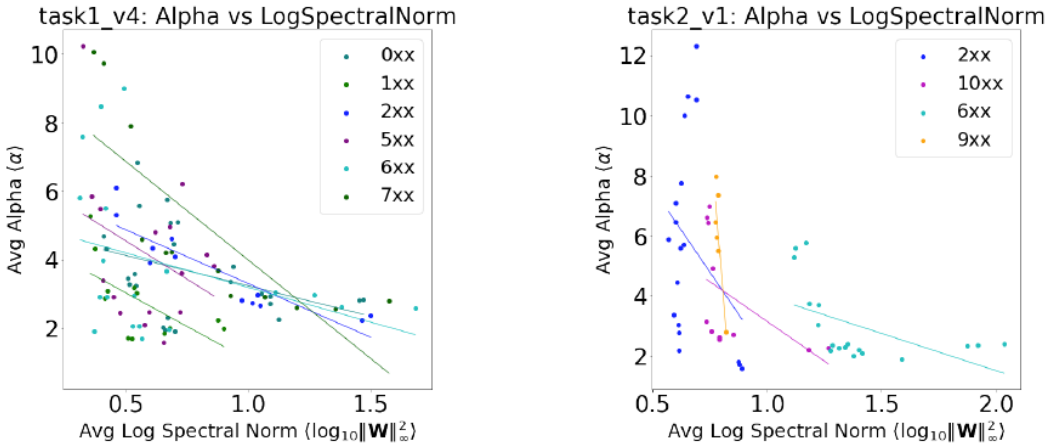
Best-performing metrics.

Complexity Metric	Average	Ref.	Need access to data?	Need access to initial weights?	Need access to GPUs?
`Alpha`	$\langle\alpha\rangle$	(here)	No	No	No
`QualityOfAlphaFit`	D_{KS}	(here)	No	No	No
`LogSpectralNorm`	$\langle\log_{10}\|\|\mathbf{W}\|\|_2^2\rangle$	([12])	No	No	No
`LogFrobeniusNorm`	$\langle\log_{10}\|\|\mathbf{W}\|\|_F^2\rangle$	([12])	No	No	No
`AlphaHat`	$\hat{\alpha}$	([7])	No	No	No
`LogAlphaShattenNorm`	$\langle\log_{10}\|\|\mathbf{W}\|\|_{2\alpha}^{2\alpha}\rangle$	([7])	No	No	No
`DistanceFromInit`	Δ_{init}	([12])	No	Yes	No
`TrainingAccuracy`	N/A	N/A	Yes	No	No
`Sharpness`	N/A	([12])	Yes	No	Yes
`SVDSmoothing`	N/A	(here)	Yes	No	No

Table 2: Overview of model quality metrics. Based on our initial analysis of Contest models, we propose and demonstrate the quality of `Alpha`, `QualityOfAlphaFit`, and `SVDSmoothing`. For several of the metrics, we refer to a recent summary paper [12] rather than original references.

Size versus shape

Size (norm) and shape (fitted HT parameters) are different . . .

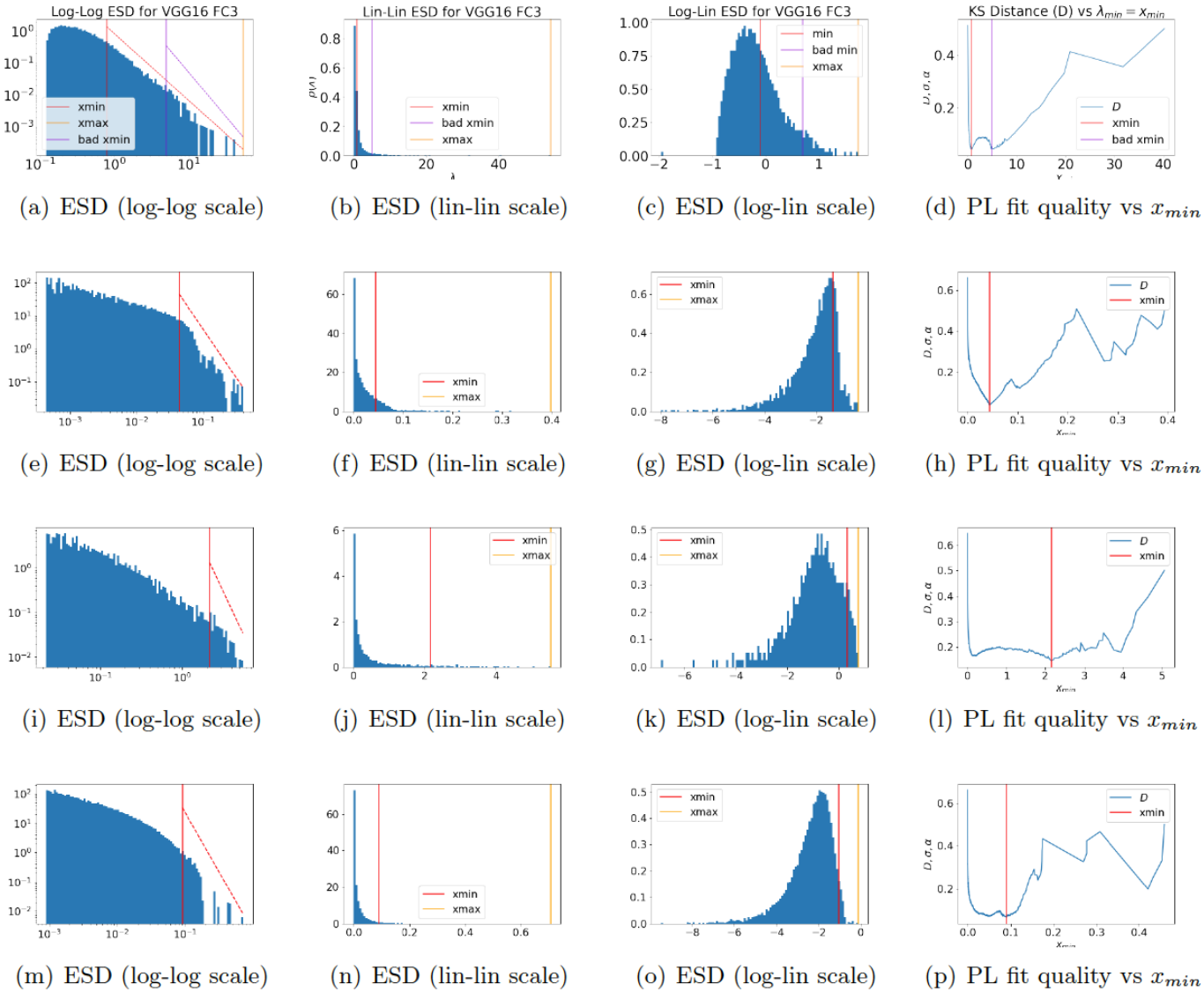


(a) `Task1` models. (b) `Task2` models.

Figure 2: Comparison of the `Alpha` and `LogSpectralNorm` metrics, for `Task1` and `Task2` models.

. . . and there is a lot of heterogeneity across tasks/subtasks.

Extracting shape parameters from HT ESDs



(a) ESD (log-log scale) (b) ESD (lin-lin scale) (c) ESD (log-lin scale) (d) PL fit quality vs x_{min}

(e) ESD (log-log scale) (f) ESD (lin-lin scale) (g) ESD (log-lin scale) (h) PL fit quality vs x_{min}

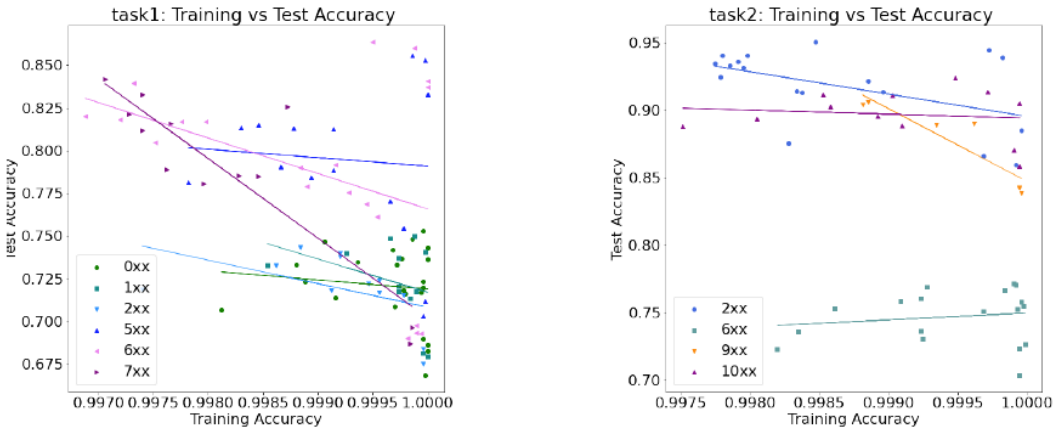
(i) ESD (log-log scale) (j) ESD (lin-lin scale) (k) ESD (log-lin scale) (l) PL fit quality vs x_{min}

(m) ESD (log-log scale) (n) ESD (lin-lin scale) (o) ESD (log-lin scale) (p) PL fit quality vs x_{min}

Figure 1: Illustration of the role of the shape of the ESD in determining the PL parameter α in

Training versus testing

Training and testing error often anti-correlated . . .



(a) `Task1` models. (b) `Task2` models.

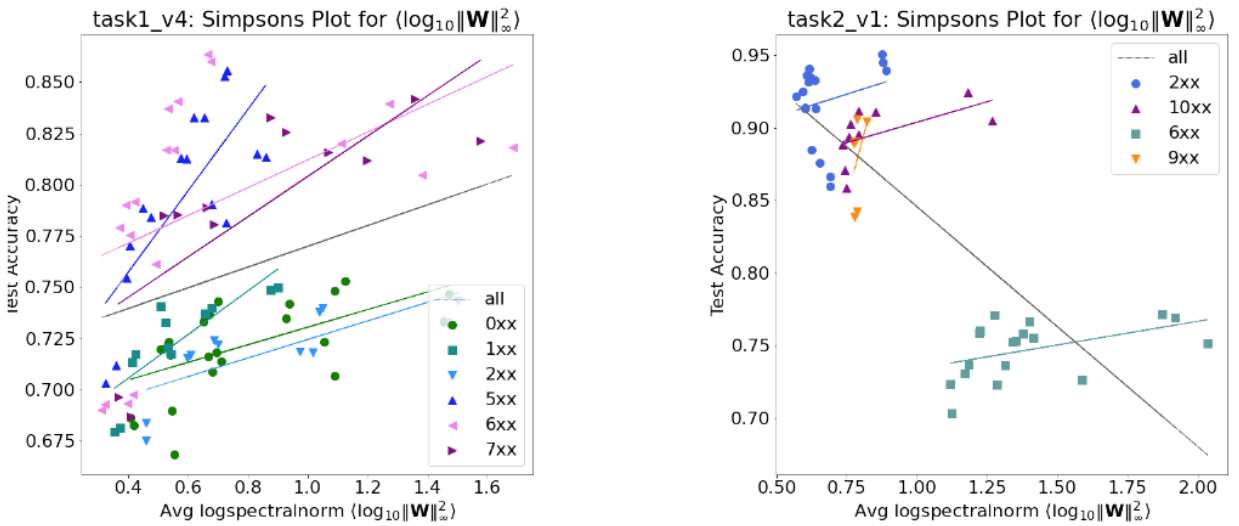
Figure 5: Relationship between training accuracy and testing accuracy for `Task1` and `Task2` models. One would expect a positive correlation or (if the training error is very close to zero) at least not a negative correlation. In many cases, they are strongly anti-correlated. See also Table 4.

. . . and there is a lot of heterogeneity across tasks/subtasks.

Simpson's paradox (1 of 2)

Within sub-group: vary solver parameters.

Between sub-groups: vary architecture.



(a) `Task1`. (b) `Task2`.

Figure 6: Illustration of Simpson's paradox. Test accuracy versus `LogSpectralNorm`, for `Task1` and `Task2`, segmented by model group. Note the overall trend is downward (line not explicitly shown), while the trend for each subgroup is upward. This is especially prominent for `Task2`.

LogSpectralNorm for better models is:

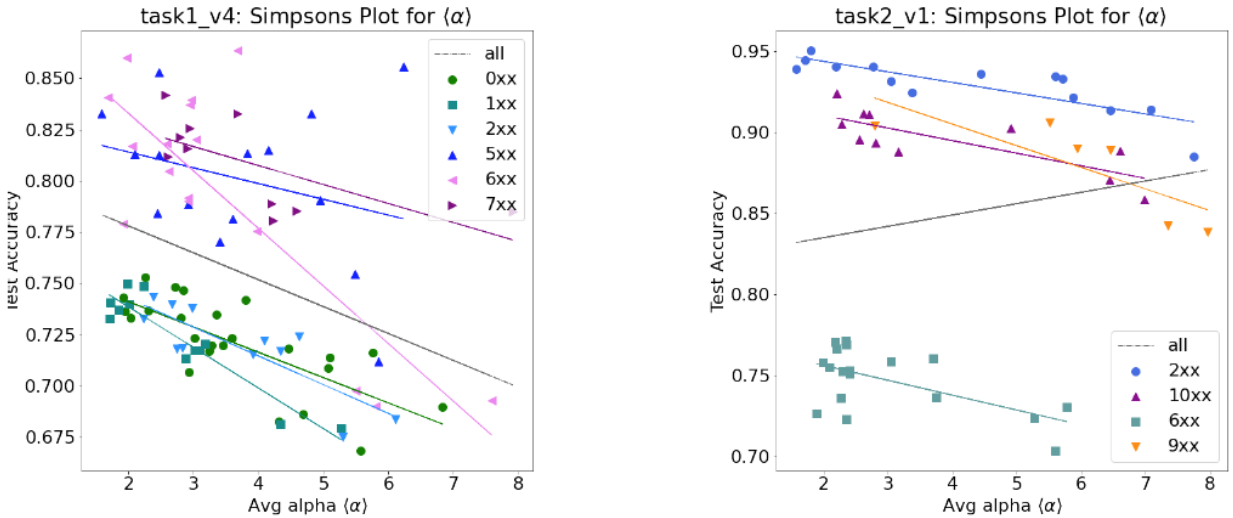
Task1: larger within *and* between sub-groups.

Task2: larger within—*and smaller between*—sub-groups.

Simpson's paradox (2 of 2)

Within sub-group: vary solver parameters.

Between sub-groups: vary architecture.



(a) `Task1`. (b) `Task2`.

Figure 7: Illustration of Simpson's paradox. Test accuracy versus `Alpha`, for `Task1` and `Task2`,

Alpha for better models is:

Task1: smaller within *and* between sub-groups.

Task2: smaller within sub-groups—*but larger between sub-groups*.

Lessons learned ...

Extracting causal insight?

- Don't invent causal metrics.
- Don't look for "one size fits all" metric.
- We identified Simpson's paradoxes—and then we used them and domain knowledge to identify causes of good performance.
- A cautionary tale . . .

Size versus shape more generally:

- Construct data-dependent versions of size versus shape.
- SVDSmoothing—if training data fit exactly, feed data through low-rank approximation. (No GPUs!)

What more can we do?

Future directions (*all of which demand a practical theory*):

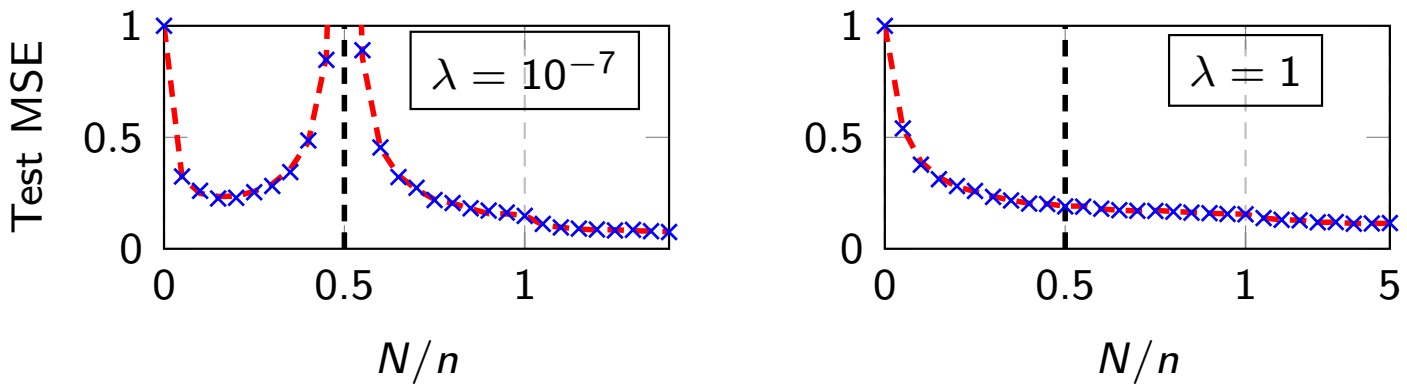
- Training/testing curves gives limited insight:
  - don't take into account hyperparameter fiddling;
  - don't correlate with robustness/accuracy/fairness/etc.
- No access to data / optimization protocols / hyperparameter values / etc.:
  - can I evaluate systems-motivated model adjustments?
  - batch size, edge, distillation, etc. (without training/retraining)?
- Model user is not a model developer:
  - sanity check: did you give me a bad/damaged model?
  - robustness check: can I look for backdoor adversarial attacks, etc.?
- Data costs money:
  - Do I have enough data?
  - Should I spend money on analysts or machines or data?

If AI/ML is to become an industrial process, beyond FAAMG, it will have to be compartmentalized to scale: Group-A develops; Group-B validates; and Group-C deploys

Outline

Data-dependent Theory of Over-param with RMT: Phase Transition and Double Descent *with Zhenyu Liao and Romain Couillet*

- Random Fourier features $\mathbf{\Sigma} = [\cos(\mathbf{WX}); \sin(\mathbf{WX})] \in \mathbb{R}^{2N \times n}$ of data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{p \times n}$ with standard Gaussian $\mathbf{W} \in \mathbb{R}^{N \times p}$
- [RR08]: entry-wise convergence of RFF Gram $\frac{1}{N}[\mathbf{\Sigma}^\top \mathbf{\Sigma}]_{ij} \to [\mathbf{K}_{\text{Gauss}}]_{ij}$ Gaussian kernel matrix, a.s. as $N \to \infty$ (pf: LLN)
- **NOT true** in spectral norm, $\|\mathbf{\Sigma}^\top \mathbf{\Sigma}/N - \mathbf{K}_{\text{Gauss}}\| \not\to 0$ unless $2N \gg n$
  - due to $\|\mathbf{A}\|_\infty \leq \|\mathbf{A}\| \leq n\|\mathbf{A}\|_\infty$
  - double descent test curves on real-world data? Yes, **proved** for RFF!
  - direct consequence of phase transition from under- to over-param of the resolvent $\mathbf{Q} = (\frac{1}{n}\mathbf{\Sigma}^\top \mathbf{\Sigma} + \lambda \mathbf{I})^{-1}$ in the ridgeless $\lambda \to 0$ limit



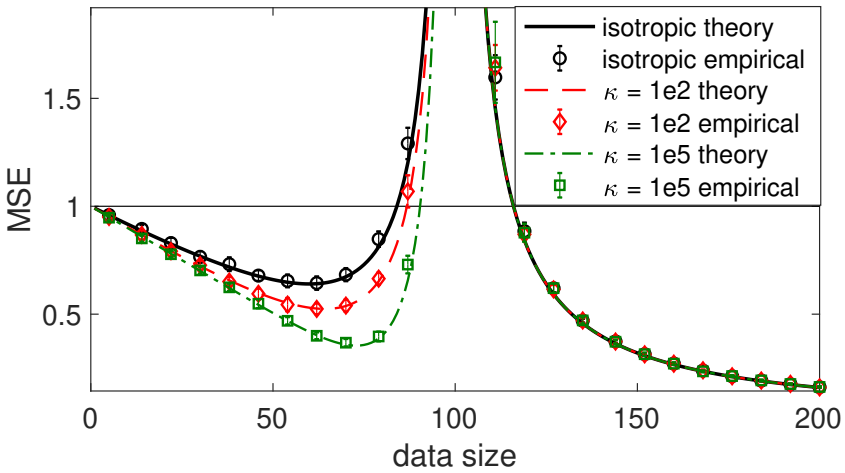
Exact expressions for double descent and implicit regularization

with Michał Dereziński and Feynman Liang

Double descent

"Classical" ML: *parameters* $\ll$ *data*
"Modern" ML: *parameters* $\gg$ *data*
Phase transition: *parameters* $\sim$ *data*

Our contribution:
New *exact* analysis for a linear model



Implicit regularization

Why does "Modern" ML work?
Because it induces implicit regularization

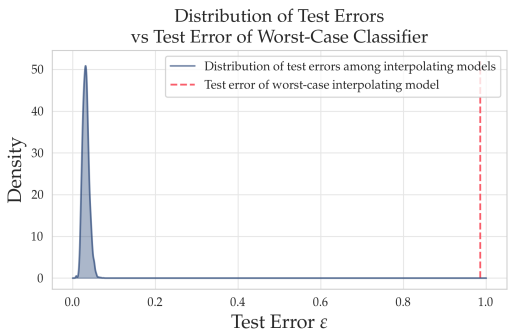
Our contribution:
Implicit *ridge* regularization of the minimum-norm solution $\mathbf{X}^\dagger\mathbf{y}$

$$\mathbb{E}[\mathbf{X}^\dagger\mathbf{y}] \simeq \text{argmin}_{\mathbf{w}}\mathbb{E}\left[(\mathbf{x}^\top\mathbf{w} - y)^2\right] + \overbrace{\lambda\,\|\mathbf{w}\|^2}^{\text{ridge}}$$

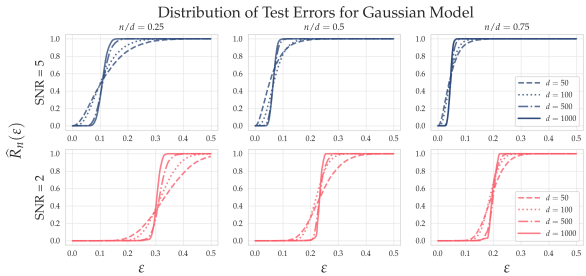
when parameters $\gg$ data

Good classifiers are abundant in the interpolating regime

with Ryan Theisen and Jason Klusowski



Most classifiers have test error much smaller than the worst-case classifier.



As dimension d gets large, with n/d fixed, test errors concentrate around a critical value ε^*.

Worst case versus average/typical case learning
Classical *uniform convergence* approach to learning studies worst-case model we could fit:

$$\varepsilon_{\text{unif}} = \sup_f \text{Test Error}(f)$$

How likely are we to actually observe the worst-case?
We develop a methodology to compute the full distribution of test errors for interpolating binary classification models:

$$R_n(\varepsilon) = \mathbb{P}(\text{Test Error}(f) \leq \varepsilon \mid f(x_i) = y_i \ \ \forall i = 1, \ldots, n)$$

Main conclusions:

- An overwhelming proportion of interpolating models have very small test error, even though worst-case models do exist.
- As model sizes grow, test errors concentrate sharply around a critical value ε^*.

Multiplicative noise and heavy tails in stochastic optimization

with Liam Hodgkinson

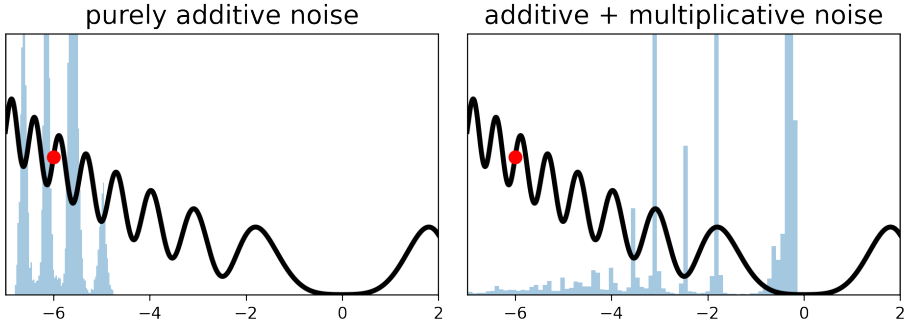
Types of noise

Iterations of stochastic gradient descent often behave like

$$W_{k+1} = W_k - A_k \nabla f(W_k) + B_k,$$

- multiplicative noise (A_k);
- additive noise (B_k)

What role does multiplicative noise play?



Our Findings

Multiplicative noise results in heavy-tailed fluctuations

$$\mathbb{P}(\|W_{k+1} - W_k\| > t) \sim ct^{-\alpha}$$

- Investigate dependence of α on model, data, and optimizer
- Theory applies to other optimizers (e.g. Newton, Adam)

Heavier tails $\implies$ **improved exploration**

Figure: Histogram of iterates on **1D non-convex objective** & initial point.

Hessian information at scale[¶]: pyHessian and ADAHessian

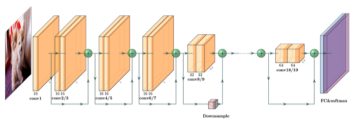
with Amir Gholami, Zhewei Yao, etc.

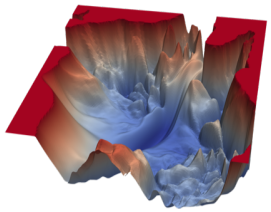
PyHESSIAN

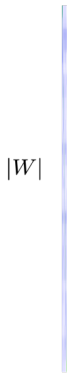
$$\min_w E(w) = \frac{1}{N}\sum_{i=1}^{N} cost(w, x_i)$$

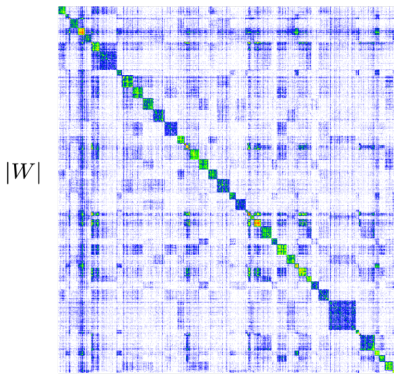
Gradient: $\frac{\partial E}{\partial w} \in \mathcal{R}^{|W|}$

Hessian: $\frac{\partial^2 E}{\partial w^2} \in \mathcal{R}^{|W|\times|W|}$









Compute lots of Hessian information for:

- Training (ADAHESSIAN)
- Quantization (HAWQ, QBERT, I-BERT)
- Pruning
- Inference

PyHessian is a pytorch library for Hessian based analysis of neural network models. It enables computing:

- Top Hessian eigenvalues
- The trace of the Hessian matrix
- The full Hessian Eigenvalues Spectral Density (ESD)

Also used for:

- Validation: loss landscape
- Validation: model robustness
- Validation: adversarial data
- Validation: test hypotheses

[¶] *TLDR: It takes 2X backprop time!*

Outline

Conclusions

"Practical theory" is not an oxymoron:

- not all theory is practical, but some is

"Practical theory" is theory for practical things:

- like data
- like SOTA DNNs

"Practical theory" can be used to address practical questions:

- is my network fully optimized?
- should I buy more data?
- can I use labels and/or domain knowledge more efficiently?
- can I design better ensembles, or improve model post-modification?
- is my pre-trained SOTA DNN overparameterized or underparameterized?

If you want more ... "pip install weightwatcher" ...