

# Building foundations for scientific machine learning at scale

Michael W. Mahoney

*Machine Learning and Analytics, SciData, LBNL  
ICSI and Department of Statistics, UC Berkeley*

March 2022

# Outline

## Introduction and Overview

Methods inform science: Learning meaningfully continuous scientific systems (Aditi Krishnapriyan, etc.)

Science informs methods: A phenomenological theory for SOTA NN performance (Charles Martin, etc.)

Implementations at scale: Putting randomized matrix algorithms within LAPACK (RandLAPACK) (Riley Murray, etc.)

Conclusions

# Foundations of data?

- ▶ NSF's Transdisciplinary Research in Principles of Data Science (TRIPODS) program
  - ▶ integrate three areas central to the foundations of data by uniting the statistics, mathematics, and theoretical computer science research communities.
- ▶ UC Berkeley FODA (Foundations of Data Analysis) Institute
  - ▶ complexity theory of inference via optimization
  - ▶ stability as a computational-inferential principle
  - ▶ randomness as a statistical-algorithmic resource
  - ▶ *principled combination of science-based and data-driven models*

# What did AlphaFold learn?

## Article

# Highly accurate protein structure prediction with AlphaFold


<https://doi.org/10.1038/s41586-021-03819-2>

Received: 11 May 2021

Accepted: 12 July 2021

Published online: 15 July 2021

Open access

 Check for updates

John Jumper<sup>1,4</sup>, Richard Evans<sup>1,4</sup>, Alexander Pritzel<sup>1,4</sup>, Tim Green<sup>1,4</sup>, Michael Figurnov<sup>1,4</sup>, Olaf Ronneberger<sup>1,4</sup>, Kathryn Tunyasuvunakool<sup>1,4</sup>, Russ Bates<sup>1,4</sup>, Augustin Židek<sup>1,4</sup>, Anna Potapenko<sup>1,4</sup>, Alex Bridgland<sup>1,4</sup>, Clemens Meyer<sup>1,4</sup>, Simon A. A. Kohl<sup>1,4</sup>, Andrew J. Ballard<sup>1,4</sup>, Andrew Cowie<sup>1,4</sup>, Bernardino Romera-Paredes<sup>1,4</sup>, Stanislav Nikolov<sup>1,4</sup>, Rishub Jain<sup>1,4</sup>, Jonas Adler<sup>1</sup>, Trevor Back<sup>1</sup>, Stig Petersen<sup>1</sup>, David Reiman<sup>1</sup>, Ellen Clancy<sup>1</sup>, Michal Zielinski<sup>1</sup>, Martin Steinegger<sup>2,3</sup>, Michalina Pacholska<sup>1</sup>, Tamas Berghammer<sup>1</sup>, Sebastian Bodenstein<sup>1</sup>, David Silver<sup>1</sup>, Oriol Vinyals<sup>1</sup>, Andrew W. Senior<sup>1</sup>, Koray Kavukcuoglu<sup>1</sup>, Pushmeet Kohli<sup>1</sup> & Demis Hassabis<sup>1,4</sup>

Proteins are essential to life, and understanding their structure can facilitate a

- ▶ ML vs ML *for* science vs ScientificML
- ▶ Goals of ML for industry vs goals of ML for science: tension bw identifying patterns in data versus discovering patterns in the world from data.
- ▶ *What does AlphaFold know that armies of biologists do not know?*

# Combining domain-driven and data-driven models?

---

## Characterizing possible failure modes in physics-informed neural networks

---

Aditi S. Krishnapriyan<sup>\*,1,2</sup>, Amir Gholami<sup>\*,2</sup>,  
Shandian Zhe<sup>3</sup>, Robert M. Kirby<sup>3</sup>, Michael W. Mahoney<sup>2,4</sup>

<sup>1</sup>Lawrence Berkeley National Laboratory, <sup>2</sup>University of California, Berkeley,

<sup>3</sup>University of Utah, <sup>4</sup>International Computer Science Institute

{aditik1, amirgh, mahoneymw}@berkeley.edu, {zhe, kirby}@cs.utah.edu

### Abstract

Recent work in scientific machine learning has developed so-called physics-

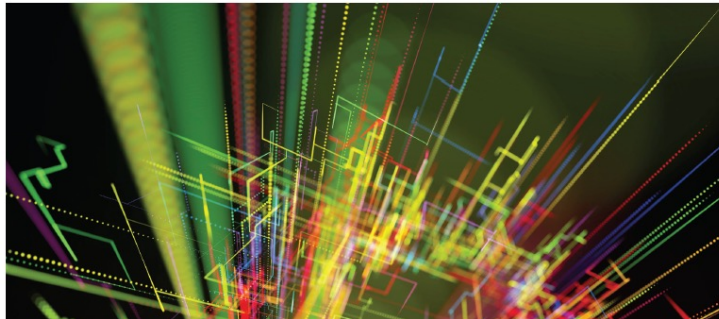
Science | DOI:10.1145/3524015

Chris Edwards

## Neural Networks Learn to Speed Up Simulations

*Physics-informed machine learning is gaining attention,  
but suffers from training issues.*

**P**HYSICAL SCIENTISTS AND engineering research and development (R&D) teams are embracing neural networks in attempts to accelerate their simulations. From quantum mechanics to the prediction of blood flow in the body, numerous teams have reported on speedups in simulation by swapping conventional finite-element solvers for models trained on various combinations of experimental and synthetic data.



# Outline

Introduction and Overview

Methods inform science: Learning meaningfully continuous scientific systems (Aditi Krishnapriyan, etc.)

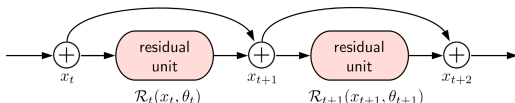
Science informs methods: A phenomenological theory for SOTA NN performance (Charles Martin, etc.)

Implementations at scale: Putting randomized matrix algorithms within LAPACK (RandLAPACK) (Riley Murray, etc.)

Conclusions

## Connection between ResNets and Dynamical Systems

- ResNets are the most popular network architectures on the market.



- Hypothesis:** Recent literature notes that ResNets learn a forward Euler discretization of a dynamical system:

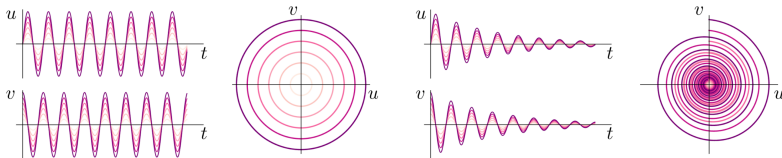
$$x_{k+1} = x_k + \Delta t \mathcal{R}(x_k, \theta_k) \longrightarrow \frac{\partial x(t)}{\partial t} = \mathcal{R}(x(t), t, \theta)$$

*=1 sneak it in*

- Spoiler: we show that ResNets are not forward Euler discretizations of a dynamical system in a meaningful way due to overfitting.

## Experiments in Dynamics

- What does it even mean to say ResNet learns a forward Euler representation of a dynamical system?
- We need context where a dynamical system is meaningful.
- So ... let's try time series prediction of a dynamical system.





# Numerical Integration and Machine Learning work in Opposite Directions

## Numerical Integration:

Ground Truth Dynamics

$$f = \frac{dx}{dt}$$

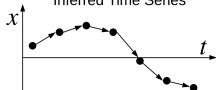


Approximation yields a model

$$x_{n+1} = x_n + \underbrace{\Delta t f(x_n)}$$



Inferred Time Series



## Learning the Dynamics:

What's G???

$$G(x; \theta)$$

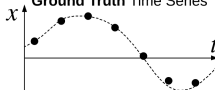


Use data to optimize a model

$$\min \|x_{n+1} - (x_n + \Delta t \underbrace{G(x_n)})\|$$



Ground Truth Time Series



## Revisiting a Simple Dynamical System

- Learning a residual makes sense in many contexts:

Future = Now + Update

- Let's study training such a  $F(x)$  based on a neural network  $G(x)$ :

$$x(t+\Delta t) = F(x(t)) = x(t) + G(x(t)) = \text{NumericalMethod}[G(x)]$$

- Numerical integrators approximate the integral with a discrete series of applications of  $f(x,t)=dx/dt$  for a time step  $\Delta t$ :

$$\begin{aligned} x(t + \Delta t) &= x(t) + \int_t^{t+\Delta t} f(x, t) dt \\ &\approx x(t) + \text{scheme}[f, x, t, \Delta t] \end{aligned}$$

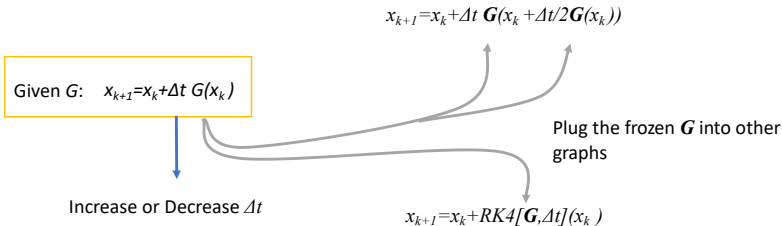
---

## Syntactic Similarity is not Sufficient for Correspondence

- Approximations to dynamical systems have richer properties.
  - A. For a given integrator, as  $\Delta t \rightarrow 0$ ,  $\text{error} \rightarrow 0$  (timestep refinement)
  - A. Integrators have a rate of convergence:  $\log(\text{error}) \propto r \log(\Delta t)$
  - A. The same  $dx/dt$  with different integrators should approach the same  $x(t_{\max})$  at their respective rates
- We can verify these using a *convergence test*.
- These conditions are critical to deriving integration schemes.  
(They also make great integration tests for numerical software!)

## Does ResNet Units Satisfies these Properties?

- If yes, then we should be able to alter the model:

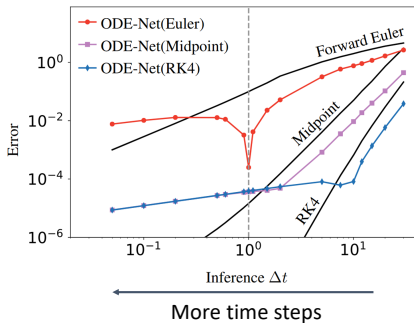


- and predictions should change consistently as expected.
- If no, then the model should behave differently w.r.t.  $\Delta t$ .

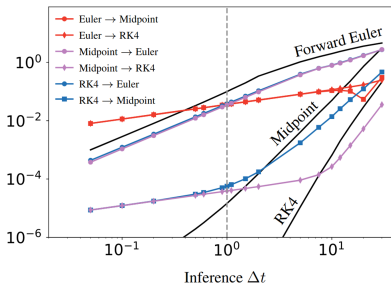
## One-off plots aren't sufficient

- Sweep  $\Delta t$  and calculate errors to do the full convergence test.

Part 1:  $\Delta t \rightarrow 0$  with same graph



Part 2: Try different integrators



# Illustration of Convergence Test for ODE-Nets

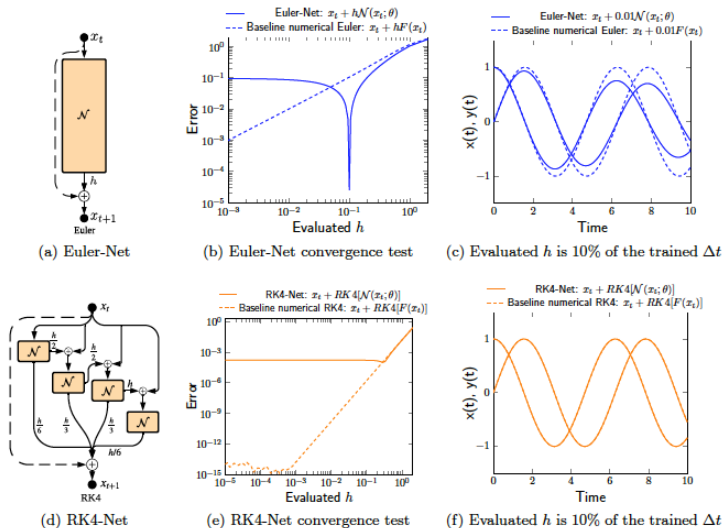


Figure 2: Illustration of convergence tests with different ODE-Nets. (a) Schematic of an ODE-Net

# Four Prototypical Dynamical Systems

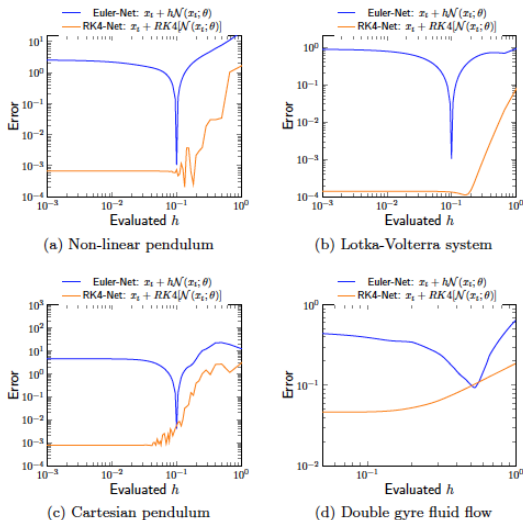


Figure 3: Illustration of convergence tests on prototypical dynamical systems. We demonstrate

# Interpolation: Predicting Fine-scale Solutions from Coarse Training Data

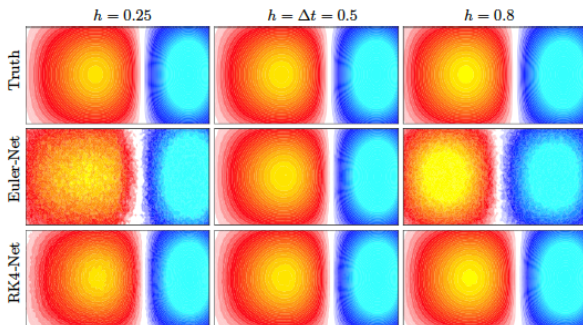
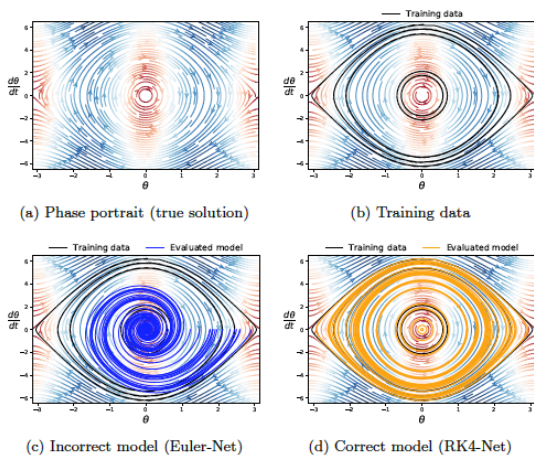


Figure 4: *Double gyre fluid flow: Reconstructing fine-scale flow fields from coarse training data.* The training data for this problem consists of vorticity field snapshots of the dynamical system taken



# Extrapolation: Predicting Trajectories for New Initial Conditions



*Figure 5: Non-linear pendulum: Extrapolation to predict initial condition trajectories on which the model was not trained. ODE-Net models are trained on randomly chosen initial conditions (different*

# Irregularly Sampled Training Data

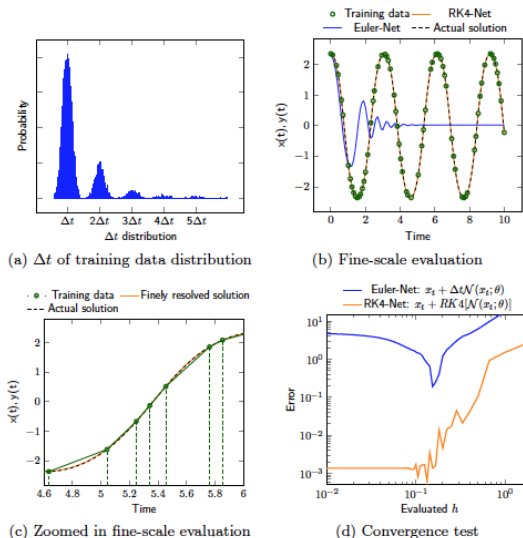


Figure 6: Learning continuous dynamics from irregularly spaced discrete points. (a) Training

# Outline

Introduction and Overview

Methods inform science: Learning meaningfully continuous scientific systems (Aditi Krishnapriyan, etc.)

Science informs methods: A phenomenological theory for SOTA NN performance (Charles Martin, etc.)

Implementations at scale: Putting randomized matrix algorithms within LAPACK (RandLAPACK) (Riley Murray, etc.)

Conclusions

# Some impracticalities

- NN training is more of a dark art than science/engineering
  - ▶ Lots of “tricks” that do not port
  - ▶ Extensive expensive hyperparameter tuning
  - ▶ ...
- NNs are nonrobust
  - ▶ Adversarial perturbations, backdoor attacks, etc.
  - ▶ Few design principles beyond training/testing “errors”
  - ▶ Not designed with counterfactual considerations
  - ▶ ...
- NNs require huge amounts of data
  - ▶ How to design models if you are data poor?
  - ▶ How to develop models if you are data poor?
  - ▶ How to deploy models if you are data poor?
  - ▶ ...

# A motivating question

~~Given a SOTA CV/NLP/Recsys model: how to tell if you have trained with enough data (e.g., what metric, with/without any data)? Is the model overtrained? Etc.?~~

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.
  - ▶ 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,

# 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.”*

# 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)

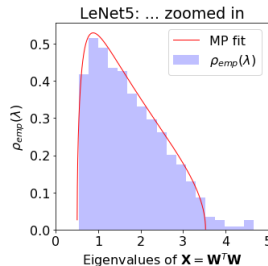
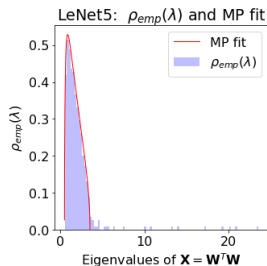


Figure: Full and zoomed-in ESD for LeNet5, Layer FC1.

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



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

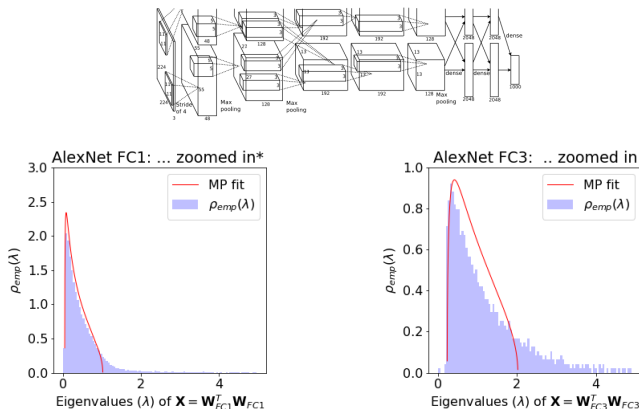
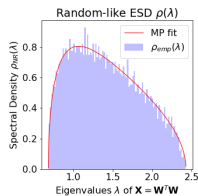


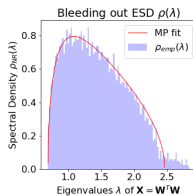
Figure: Zoomed-in ESD for Layer FC1 and FC3 of AlexNet.

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

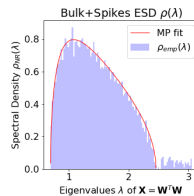
# RMT-based 5+1 Phases of Training (in pictures)



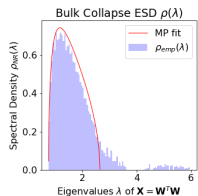
(a) RANDOM-LIKE.



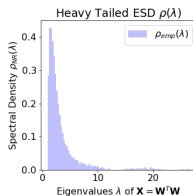
(b) BLEEDING-OUT.



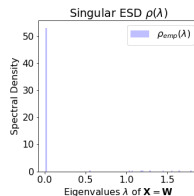
(c) BULK+SPIKES.



(d) BULK-DECAY.



(e) HEAVY-TAILED.



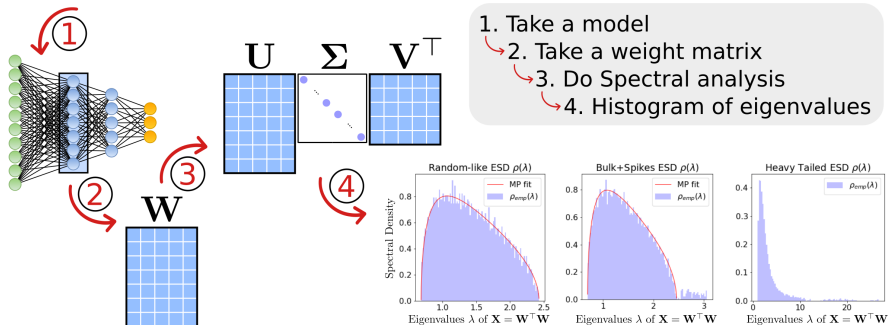
(f) RANK-COLLAPSE.

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

# Watching weights with WeightWatcher

<https://github.com/CalculatedContent/WeightWatcher>

## Analyzing DNN Weight matrices with WeightWatcher



- 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”



## Motivations: WeightWatcher Theory

*Understanding deep learning requires rethinking generalization*

*The weightwatcher theory is a Semi-Empirical theory based on:*

*the Statistical Mechanics of Generalization,  
Random Matrix Theory, and  
the theory of Strongly Correlated Systems*



## New approach: SemiEmpirical Theory

$$\lim_{N \rightarrow \infty} \frac{1}{N} \log \mathbb{E}_A \left[ \exp \left( \frac{\beta}{2} \text{Tr}[\mathbf{W}^\dagger \mathbf{A} \mathbf{W}] \right) \right] = \frac{\beta}{2} \sum_{i=1}^M G_A(\lambda_i)$$

“Asymptotics of HCZI integrals ...” [Tanaka \(2008\)](#)

“Generalized Norm”

simple, functional form  
can infer from empirical fit

Eigenvalues of Teacher  
empirical fit to:

$$G_A(\lambda) := \int_0^\lambda R_A(z) dz \quad \leftarrow R(z) = z^{\alpha-1}$$

WeightWatcher  
PowerLaw metric

$$\hat{\alpha} = \alpha \log \lambda_{max} \approx \log \sum_{i=1}^M G_A(\lambda_i)$$

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

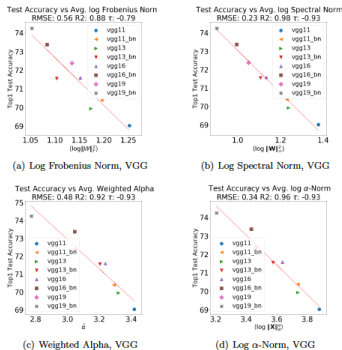


Figure 2: Comparison of Average Log Norm and Weighted Alpha quality metrics versus re test accuracy for pretrained VGG models: VGG11, VGG13, VGG16, and VGG19, with and

Different metrics on **pre-trained VGG**.

Series	#	Metric	$(\log \ \mathbf{W}\ _F^2)$	$(\log \ \mathbf{W}\ _{\infty}^2)$	$\alpha$	$(\log \ \mathbf{X}\ _2^2)$
VGG	6	RMSE	0.56	<b>0.23</b>	0.48	0.34
		$R^2$	0.88	<b>0.98</b>	0.92	0.96
		Kendall- $\tau$	-0.79	<b>-0.93</b>	-0.93	<b>-0.93</b>
ResNet	5	RMSE	0.9	0.97	<b>0.61</b>	0.66
		$R^2$	0.92	0.9	<b>0.96</b>	0.9
		Kendall- $\tau$	-1.0	-1.0	-1.0	-1.0
ResNet-1K	19	RMSE	2.4	2.8	<b>1.8</b>	1.9
		$R^2$	0.81	0.74	<b>0.89</b>	0.88
		Kendall- $\tau$	-0.79	-0.79	<b>-0.89</b>	-0.88
DenseNet	4	RMSE	0.3	<b>0.11</b>	0.16	0.21
		$R^2$	0.93	<b>0.99</b>	0.98	0.97
		Kendall- $\tau$	-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- $\tau$  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**.

	$\log \ \cdot\ _F^2$	$\log \ \cdot\ _{\infty}^2$	$\alpha$	$\log \ \cdot\ _2^2$
RMSE (mean)	4.84	5.57	4.58	4.55
RMSE (std)	9.14	9.16	9.16	9.17
$R^2$ (mean)	3.9	3.85	3.89	3.89
$R^2$ (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, *Nature Communications*, 2021.

## 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 adjacent 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<sup>§</sup> may appear, and they may even be important for model convergence.

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

---

<sup>§</sup>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.

# Hessian information at scale<sup>||</sup>: pyHessian and ADAHessian

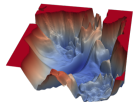
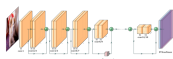
(with Amir Gholami, Zhewei Yao, etc.)

PyHESSIAN

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

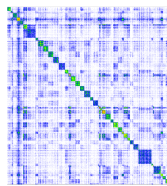
$$\text{Gradient: } \frac{\partial E}{\partial w} \in \mathcal{R}^{|W|}$$

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



|W|

|W|



|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: See our recent review: "A Survey of Quantization Methods for Efficient Neural Network Inference," by Gholami et al. (arXiv:2103.13630).

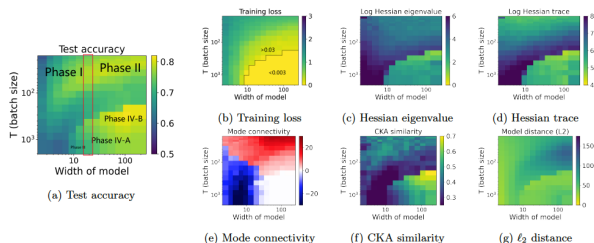
|| TLDR: It takes 2X backprop time!



# Loss landscapes and robustness

(with Yoqing Yang, etc.)

Use CKA similarity, mode connectivity, Hessian information to characterize “rugged convexity” in loss landscapes:



	Globally poorly-connected	Globally well-connected	
Locally sharp	Phase I 	Phase II 	
Locally flat	Phase III 	Phase IV-A trained models are less similar 	Phase IV-B trained models are similar 

# Outline

Introduction and Overview

Methods inform science: Learning meaningfully continuous scientific systems (Aditi Krishnapriyan, etc.)

Science informs methods: A phenomenological theory for SOTA NN performance (Charles Martin, etc.)

**Implementations at scale: Putting randomized matrix algorithms within LAPACK (RandLAPACK) (Riley Murray, etc.)**

Conclusions

# What is LAPACK?

## LAPACK (Linear Algebra PACKage)

- standard software library for numerical linear algebra
- routines for systems of linear equations and linear least squares, eigenvalue problems, and SVD
- also routines to implement associated matrix factorizations, LU, QR, Cholesky and Schur, etc.

“If you call a linear algebra routine in python, R, etc. ... then you probably call something that calls something that calls LAPACK.”

## BLAS (Basic Linear Algebra Subprograms)

- a specification that prescribes a set of low-level routines for common linear algebra operations
- vector addition, scalar multiplication, dot products, linear combinations, and matrix multiplication
- the de facto standard low-level routines for linear algebra libraries

# Randomized numerical linear algebra (RNLA)

Using randomized algorithms to solve deterministic problems.

For example:  $\min_x \|Ax - b\|_2^2$

The algorithms use randomness *internally*

- Rely on a black-box random number generator.
- The generator needn't be very high-quality.

The algorithms gamble with **solution quality** and/or **computational cost**

- Quality and cost vary from one run to another.
- Many RNLA algorithms have extremely small variations in performance.

Reviews from different perspectives: [17, 12, 18, 19, 20, 21, 22, 13, 23]

# What does randomization buy us?

- Efficient algorithms for computing **approximate solutions**

*Whole areas.* E.g., low-rank approximation [12], convex optimization [24].

- Efficient algorithms for computing **machine-precision solutions**

*Specific problems.* E.g., strongly overdetermined least squares [9, 10, 11], block column-pivoted QR [25, 26].

- Robust algorithms for **intractable problems**

E.g., nonnegative matrix factorization [27], interpolative decomposition [28]

- Solving problems under data-privacy constraints [29, 30, 31, 32]

Reviews from different perspectives: [17, 12, 18, 19, 20, 21, 22, 13, 23]

# Two ingredients of RNLA algorithms

## Random sketching

For overdetermined least squares with data  $(\mathbf{A}, \mathbf{b})$ , obtain *sketched* data

$$\hat{\mathbf{A}} = \mathbf{S} \mathbf{A}$$

and  $\hat{\mathbf{b}} = \mathbf{S} \mathbf{b}$ .

## High-level deterministic NLA

Next, solve the sketched problem

$$\min_{\mathbf{x}} \|\mathbf{S}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2.$$

For example, by SVD

$$\begin{aligned} \hat{\mathbf{A}} &= \mathbf{U} \mathbf{\Sigma} \mathbf{V}^\top, \\ \Rightarrow \hat{\mathbf{x}} &= \mathbf{V} \mathbf{\Sigma}^\dagger \mathbf{U}^\top \hat{\mathbf{b}}. \end{aligned}$$

# An architecture in two parts

Randomized LAPACK will be written in C++ and build on LAPACK++.

- Configurable object-oriented API and simplified procedural API
- Data model to focus on dense matrices in shared-memory.
- Accommodate sparse/abstract matrices with “linear operator” objects.

The Randomized BLAS will handle sketching dense data matrices.

- Procedural API *only*
- Hide all details of the random number generator (but preserve reproducibility)
- Support sketching operators drawn from a variety of distributions
- Opportunities to reorganize computation for big performance gains

# Levels in the Randomized BLAS

**Levels 1 – 3 produce sketches.** Possible organizations:

- One *sketch* at Level 1, two at Level 2, three or more at Level 3.
- One *sketching operator* at Level 1, two at Level 2, three or more at Level 3.

Special examples	$(AS, A^T AS)$	$(SA, Sb)$	$S_1 AS_2$
Level by # sketches	2	2	1
Level by # operators	1	1	2

**Level 0:** generate defining data for a sketching operator.



# Least squares problems . . . and optimization

Data matrix  $\mathbf{A}$  is  $m \times n$  and *tall* ( $m \gg n$ ).

Overdetermined least squares

$$\min_{\mathbf{x} \in \mathbb{R}^n} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

Underdetermined least squares

$$\min_{\mathbf{y} \in \mathbb{R}^m} \|\mathbf{y}\|_2^2 \quad \text{subject to} \quad \mathbf{A}^T \mathbf{y} = \mathbf{c}.$$

Randomized LAPACK:

- take a “primal-dual” perspective on these problems.
- include methods for solving to any desired accuracy.
- facilitate more general second-order optimization algorithms.

# A saddle point perspective

Consider a simple *saddle point system*

$$\begin{bmatrix} I & A \\ A^\top & 0 - \textcolor{red}{H} \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}. \quad (1)$$

Equation 1 (with  $\textcolor{red}{H} = 0$ ) characterizes optimal solutions to the *primal-dual pair*

$$\begin{aligned} \min_{x \in \mathbb{R}^n} & \|Ax - b\|_2^2 + 2c^\top x \\ \min_{y \in \mathbb{R}^m} & \|y - b\|_2^2 \text{ subject to } A^\top y = c. \end{aligned}$$

Encounter [sequences of saddle point systems](#) in ...

- $\ell_p$  regression for  $p \in (1, 2)$
- minimizing a composite convex function via Newton's method
- Interior-point methods for [quadratic](#) linear programming [when  \$\textcolor{red}{H}\$  is psd](#)

# Overdetermined least-squares example

Fixed data matrix:

$$\mathbf{A} \in \mathbb{R}^{100,000 \times 2,000}$$

$$\text{cond}(\mathbf{A}) = 100,000$$

Fixed target vector:

$$\|\mathbf{A}\mathbf{A}^\dagger \mathbf{b}\|_2 = 0.95 \|\mathbf{b}\|_2$$

LAPACK time in seconds:

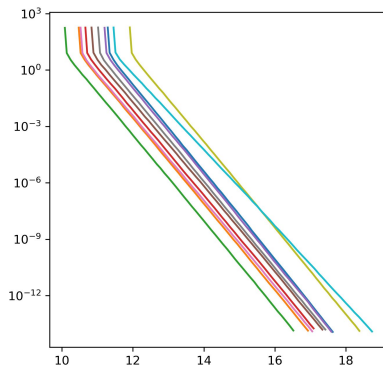
GELSD: 26.3

GELSS: 45.6

Laptop w/ Core i7-1065G7

Battery power

Normal equation error vs time in seconds



Ten trials with SRCT  $\mathbf{S} \in \mathbb{R}^{6,000 \times 100,000}$

# Low-rank approximation

*Produce a suitably factored representation of a low-rank matrix  $\hat{\mathbf{A}}$ , which stands in as an approximation for a target matrix  $\mathbf{A}$ .*

How to measure the quality of an approximation?

- Distance from the target  $\|\mathbf{A} - \hat{\mathbf{A}}\|$
- Distance from an “optimal” approximation  $\|\mathbf{A}_\star - \hat{\mathbf{A}}\|$ .

Algorithms in Randomized LAPACK

- can accept parameter  $k$ , produce  $\hat{\mathbf{A}}$  where  $\text{rank } \hat{\mathbf{A}} = \min\{k, \text{rank } \mathbf{A}\}$ .
- can (in some cases!) accept  $\epsilon$  and ensure  $\|\mathbf{A} - \hat{\mathbf{A}}\| \leq \epsilon$ .
- come with theoretical guarantees for bounding  $\|\mathbf{A} - \hat{\mathbf{A}}\|$  and/or  $\|\mathbf{A}_\star - \hat{\mathbf{A}}\|$ .

# Outline

Introduction and Overview

Methods inform science: Learning meaningfully continuous scientific systems (Aditi Krishnapriyan, etc.)

Science informs methods: A phenomenological theory for SOTA NN performance (Charles Martin, etc.)

Implementations at scale: Putting randomized matrix algorithms within LAPACK (RandLAPACK) (Riley Murray, etc.)

Conclusions

# Conclusions

- ▶ Foundations of Scientific Machine Learning
- ▶ Methods inform science: Learning meaningfully continuous scientific systems
- ▶ Science informs methods: A phenomenological theory for SOTA NN performance
- ▶ Implementations at scale: Putting randomized matrix algorithms within LAPACK (RandLAPACK)