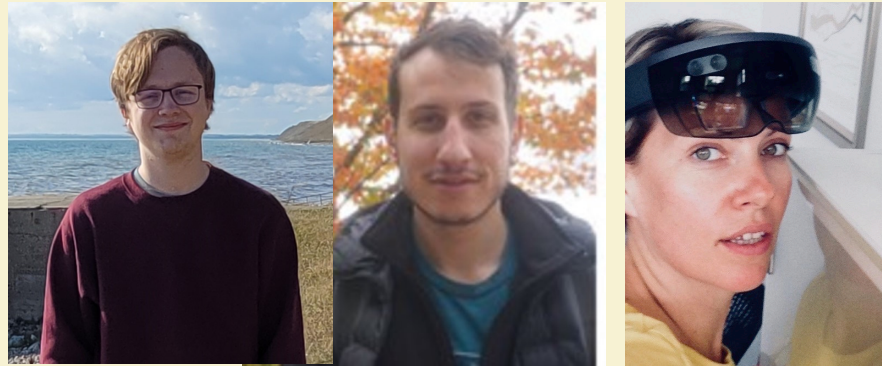


CS 229br: Foundations of Deep Learning

Lecture 6: Training Dynamics Part 2

Boaz Barak

Gustaf Ahdritz Gal Kaplun Zona Kostic



Plan:

Part I: Overview of interventions and their impacts

- Computational, optimization, and generalization efficiency
- Learning rate, batch size, normalization, preconditioning

Part II: Empirical phenomena and toy models

- Simplicity Bias, Deep bootstrap, edge of stability, scaling laws
- Kernels, Nearest-Neighbors, Depth 2 nets, linear nets

~~Theory of Deep Learning?~~ — Theory of X?



Mechanistic:

Give causal mechanisms that fully explain observed phenomena

Classical mechanics

Models:

Give fully explained toy models with qualitatively accurate predictions

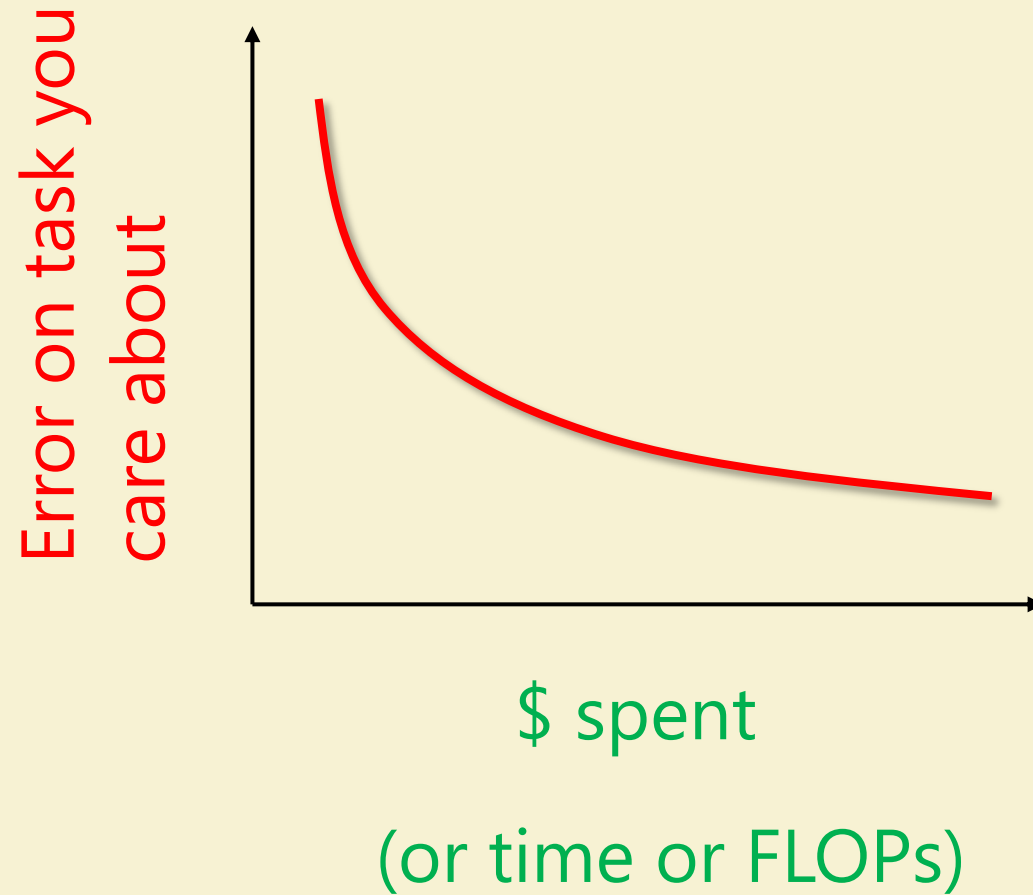
Statistical mechanics

Predictive:

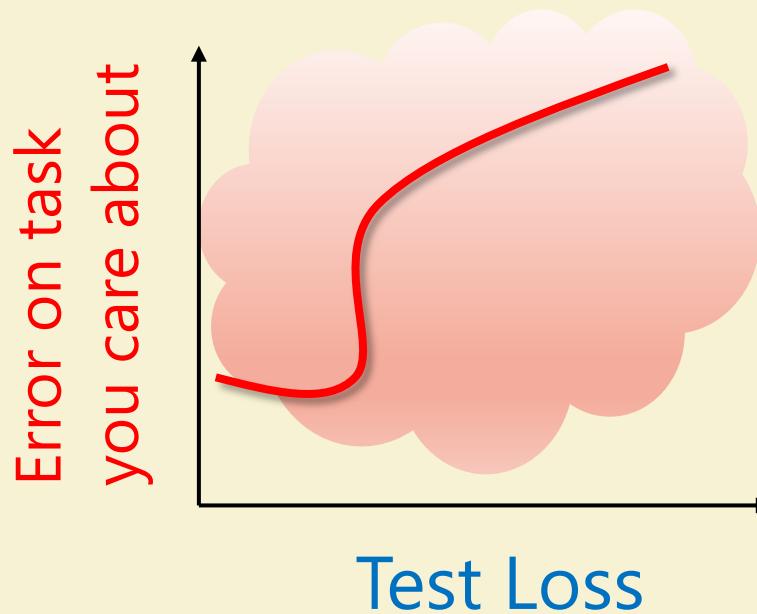
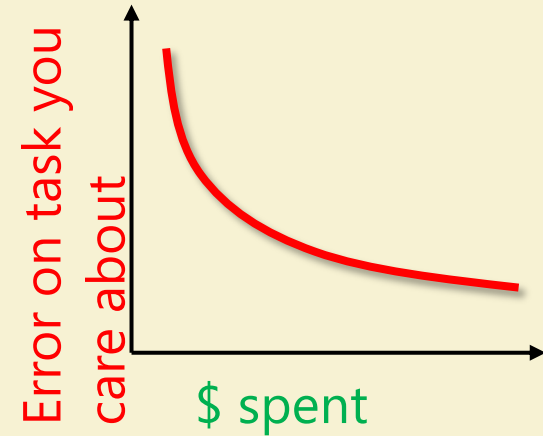
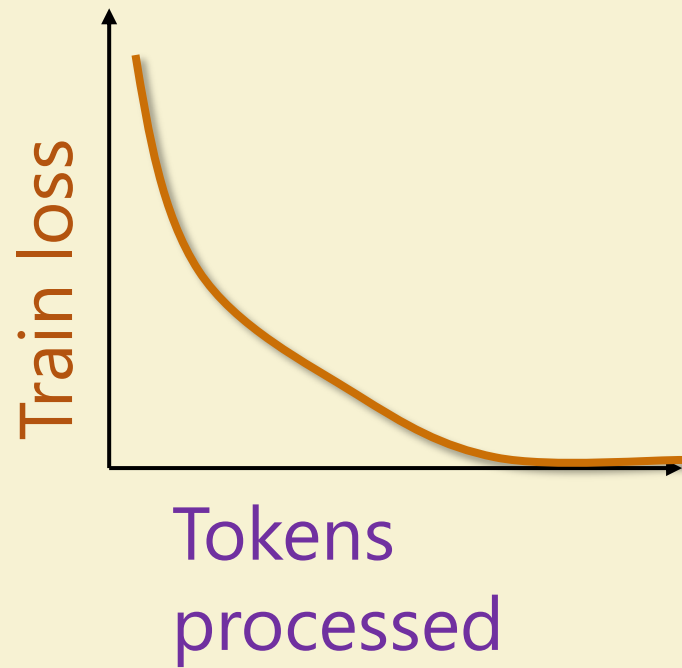
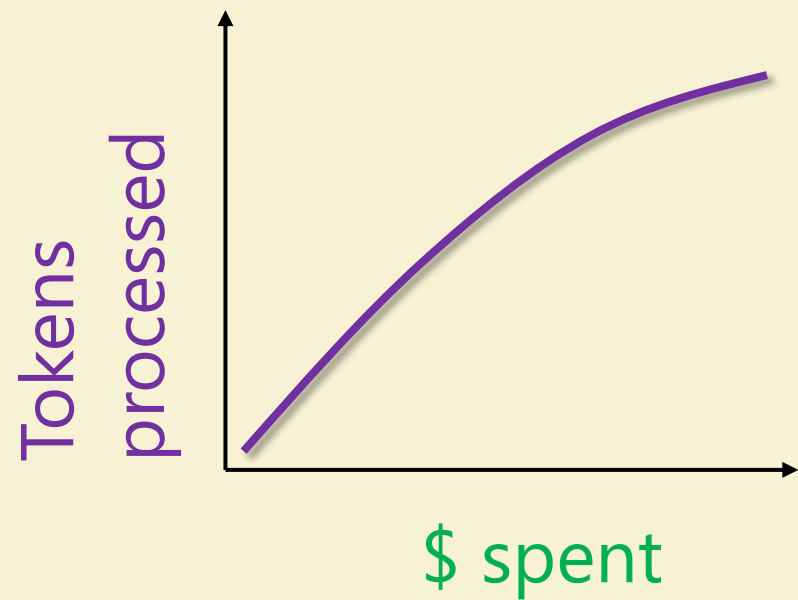
Give predictions without explaining mechanism behind them

Economics

Optimization



Acknowledgements: Roger Grosse, Horace He, Nikhil Vyas, Gustaf&Gal



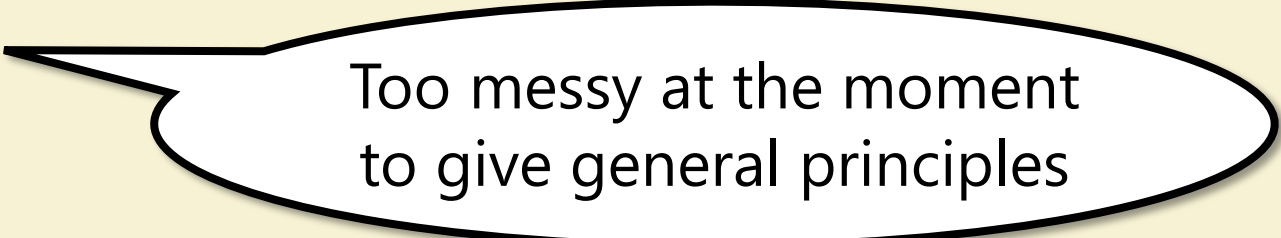
Optimization Goals

Computational Efficiency (CE): Maximize the number of tokens we can process per \$ (or per FLOP or per second)

Optimization Efficiency (OE): Maximize the reduction in training loss per tokens processed

Generalization Efficiency (GE): Maximize the reduction in validation loss per unit of reduction in train loss

Downstream Efficiency (DE): Maximize performance in downstream tasks as function of validation loss.



Too messy at the moment
to give general principles

Interventions

Increase Batch Size:

Increase number of tokens processed in parallel

Increase Learning Rate

Precondition Gradient: Apply **diagonal** (Adam), **Factored** (K-FAC, Shampoo), **general** (Natural Gradient) operation to gradient.

Weak normalization: Normalize means and magnitude of activations (layer/batch norm)

Whitening: Normalize activation covariance.

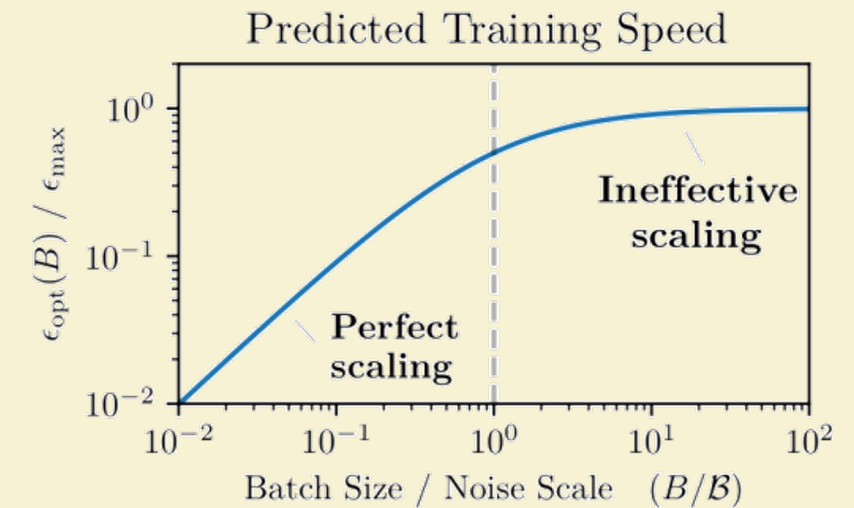
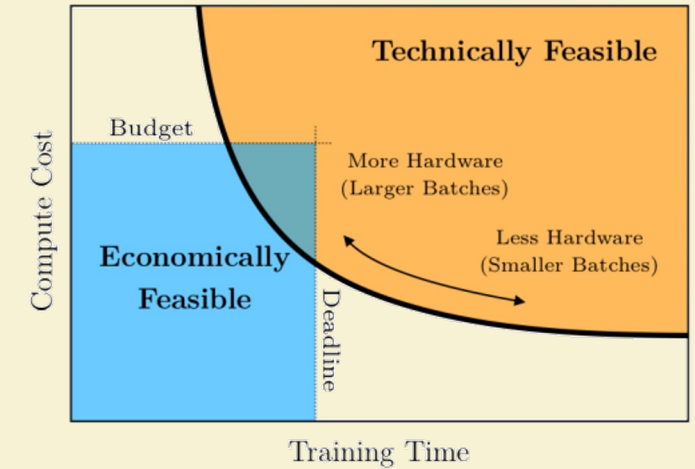
Increase Batch Size:

Increase number of tokens processed in parallel

CE: Improve wall clock time (better utilization / parallelization)

OE: No harm up to point

GE: Worse if not compensated via LR



Increase Learning Rate



LR
Schedules?

CE: neutral

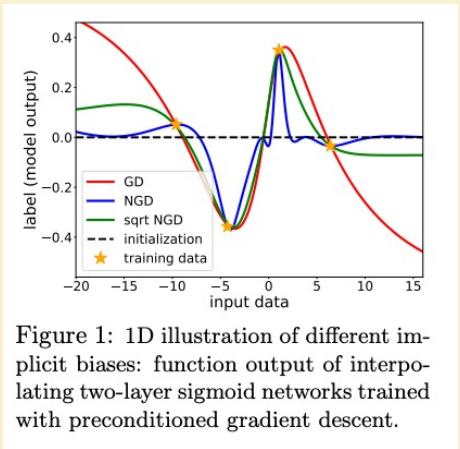
OE: Complicated – generally a “goldilocks” regime, but NN can adapt (see “edge of stability”). Sometimes lower LR could yield faster optimization at expense of generalization.

GE: Larger LR often leads to better generalization conditioned on train

Precondition Gradient: Apply **diagonal** (Adam), **Factored** (K-FAC, Shampoo), **general** (Natural Gradient) operation to gradient.

CE: From small to large overhead.

OE: Often speeds up training.



[Amari et al 20]

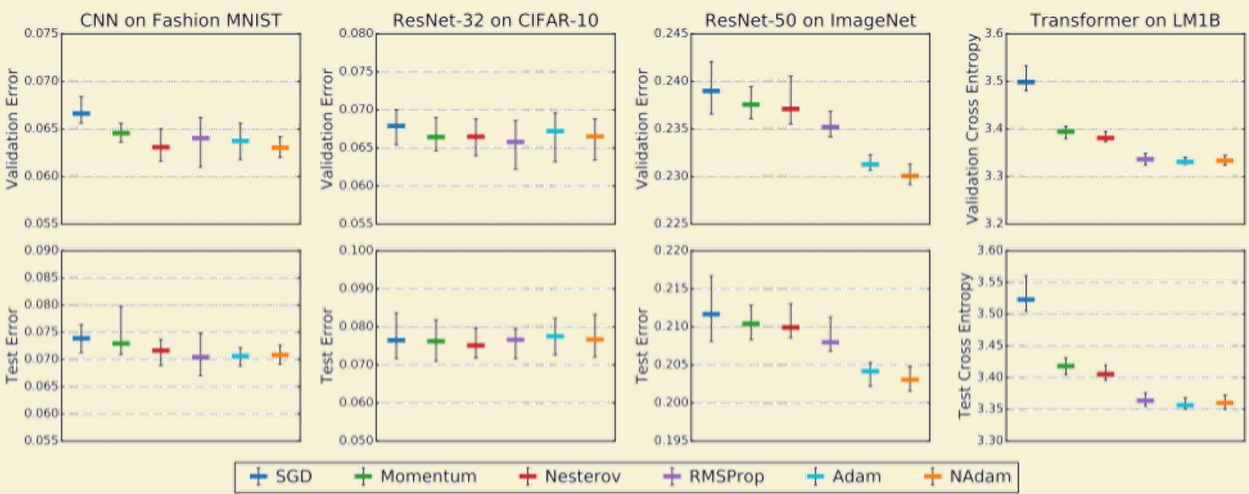
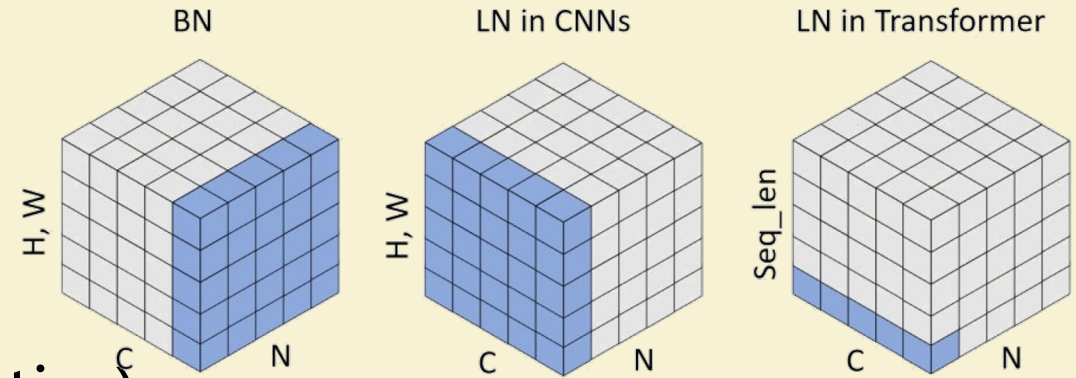


Figure 1: The relative performance of optimizers is consistent with the inclusion relationships, regardless of whether we compare final validation error (top) or test error (bottom). For all workloads, we tuned the hyperparameters of each optimizer separately, and selected the trial that achieved the lowest final validation error. Optimizers appear in the same order as the legend in all plots in this paper.

Choi et al 2019

GE: Sometimes solutions generalize worse, though can be mitigated through training params. Generally, more “knobs” to tune so can always recover SGD performance.

Weak normalization: Normalize means and magnitude of activations
(layer/batch norm)

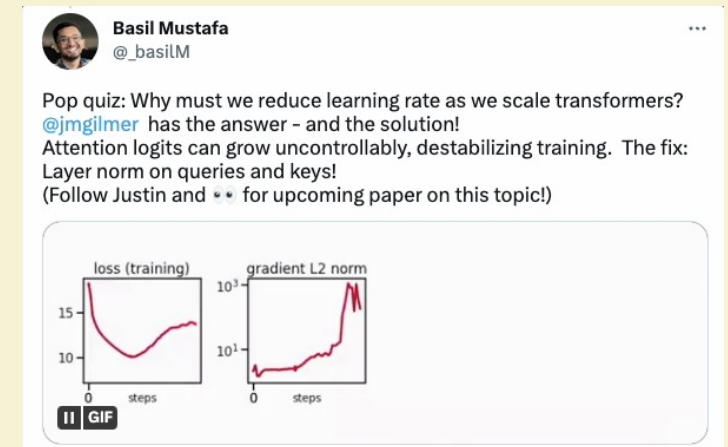


CE: Small to moderate (compute statistics)

[Yao et al, ICCV 21](#)

OE: Generally speeds up optimization

GE: Unclear (see [Brock et al 2021](#))



Whitening: Normalize activation covariance.

CE: Significant (need to compute covariances)

OE: Generally speeds up optimization

GE: Could be very negative

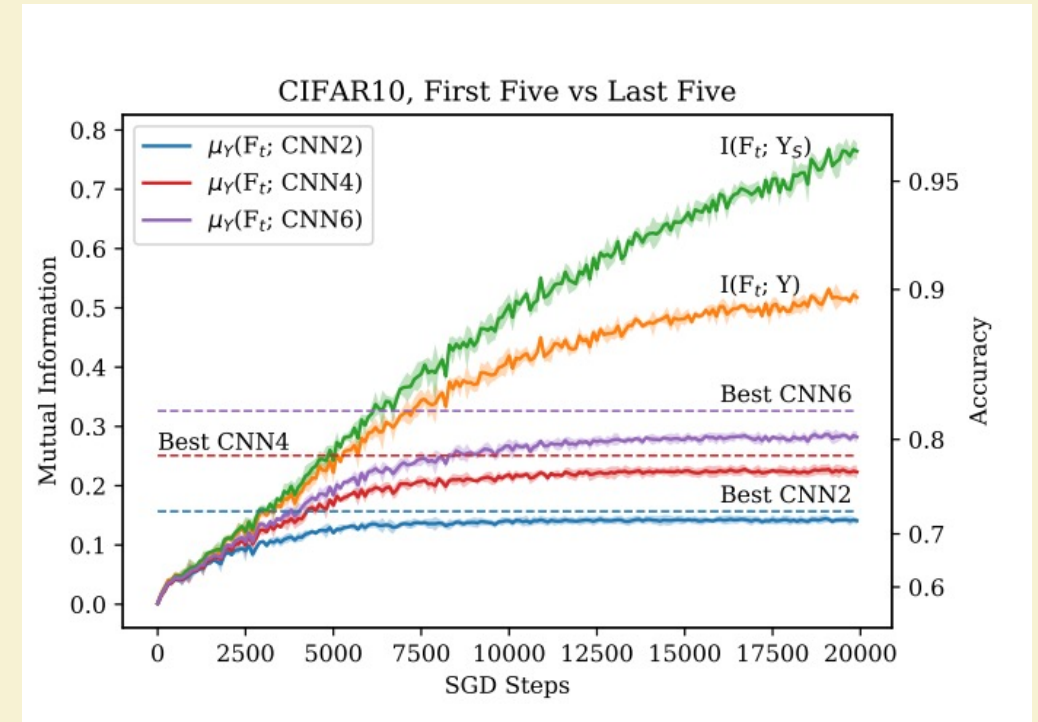
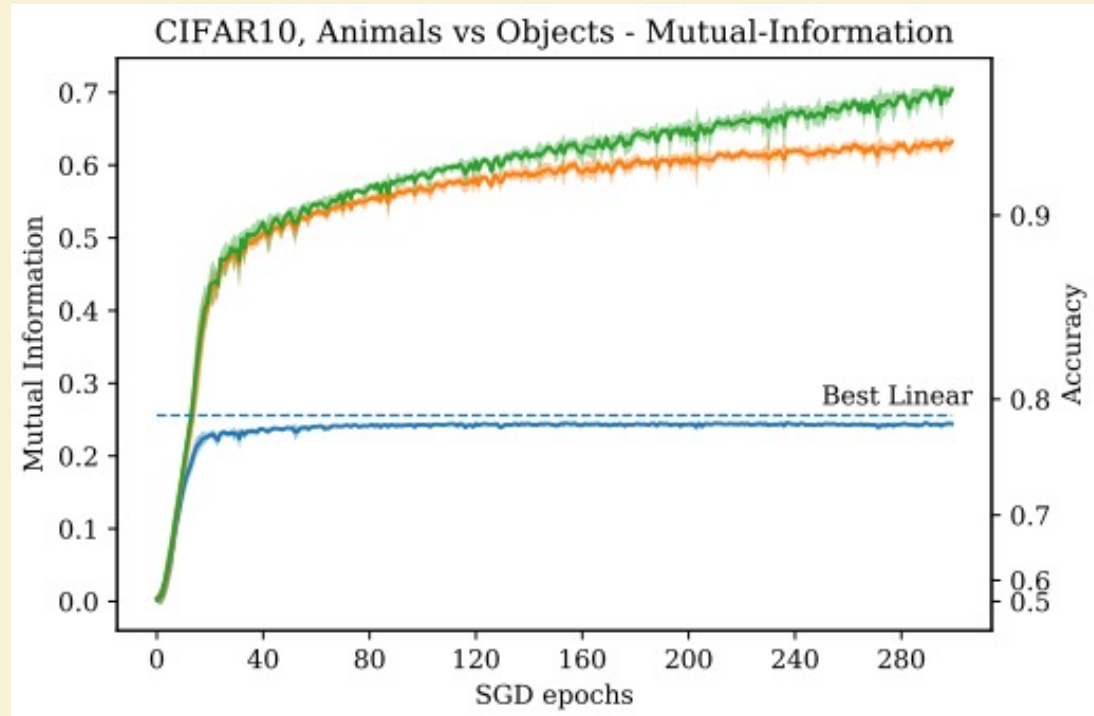
More interventions

- Momentum, exponential moving averages
- Gradient clipping
- Dropout
- More layers, wider layers
- Choosing random seed from the bible.
- Wearing your lucky underwear when training.

Part II: Optimization phenomena

- **Simplicity Bias:** SGD prefers simpler minimizers.
[\[Nakkiran et al 2019\]](#)
- **Deep Bootstrap:** Multiple epochs behave like a single one
[\[Nakkiran et al 2020\]](#)
- **Edge of stability:** The local loss surface of neural nets “progressively sharpen” and then stays on the the edge of diverging away.
[\[Cohen et al 2021\]](#)
- **Scaling laws:** Test loss curves follow somewhat predictable functional form, as function of data, model size, and computational steps.
[\[Kaplan et al 2020, Rosenfeld et al 2019, Hoffman et al 2022\]](#)

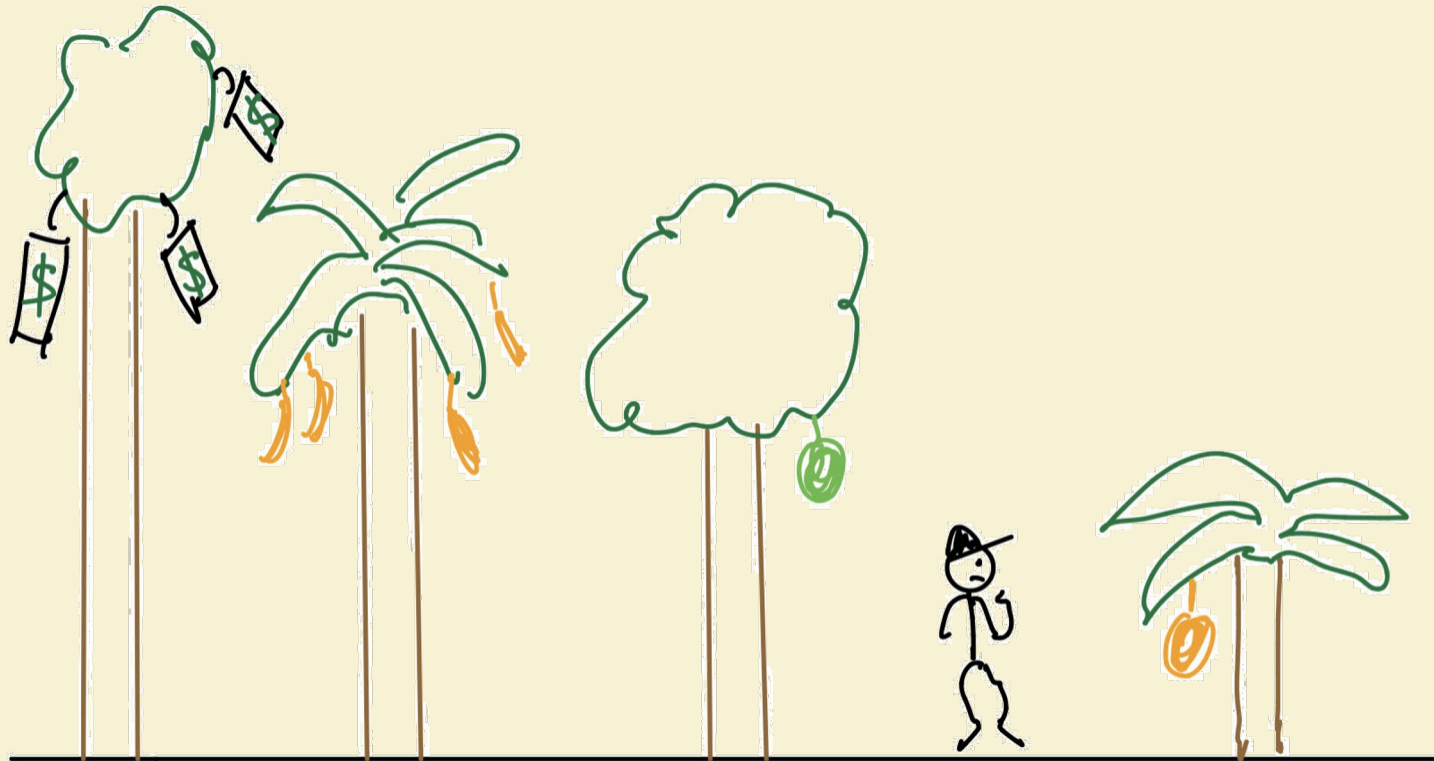
SGD Learns simple concepts first



Simplicity bias is a good thing...

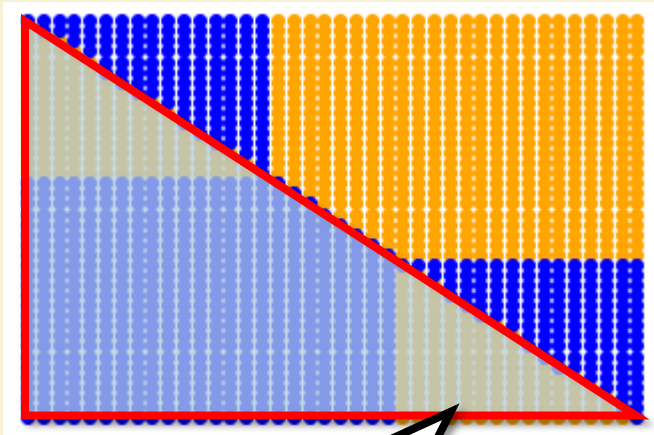
A random f fitting $(x_i, y_i)_{i=1..n}$ will never generalize.

... and a bad thing

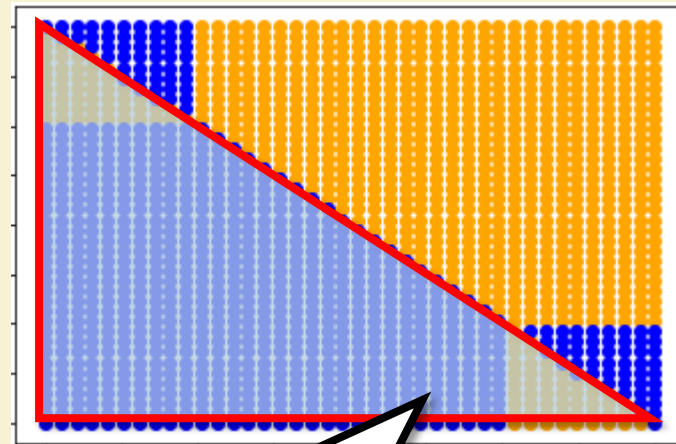


x	$f(x)$
x_1	y_1
x_2	y_2
x_3	y_3
...	...
x_n	y_n
x	—

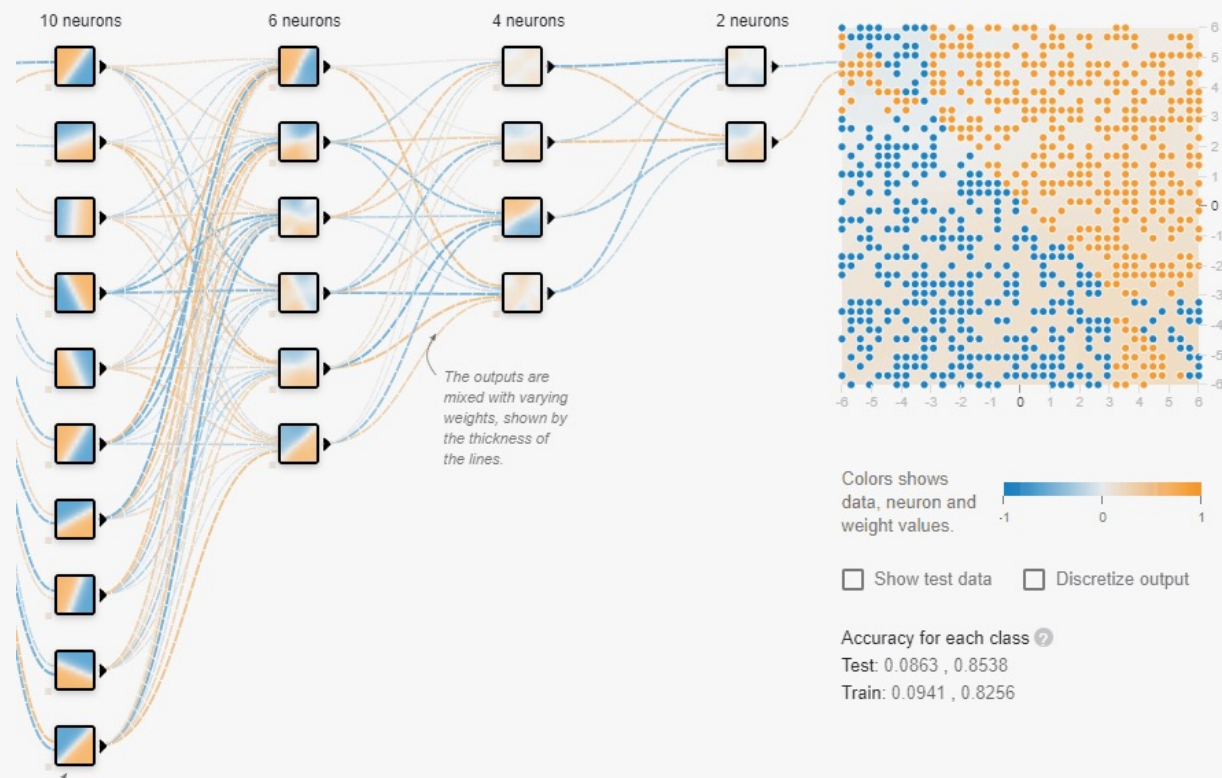
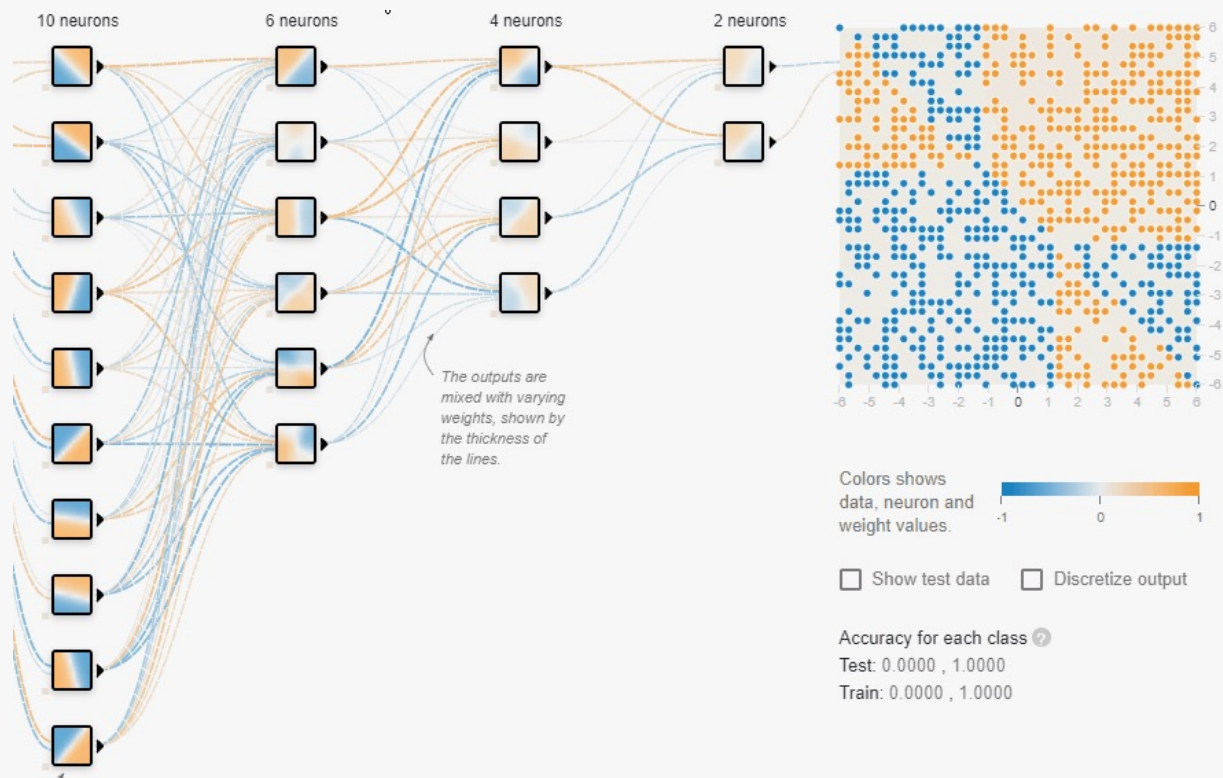
Example:



Simple solution
has **high** error



Simple solution
has **low** error



The Pitfalls of Simplicity Bias in Neural Networks

Harshay Shah
Microsoft Research
harshay.rshah@gmail.com

Kaustav Tamuly
Microsoft Research
ktamuly2@gmail.com

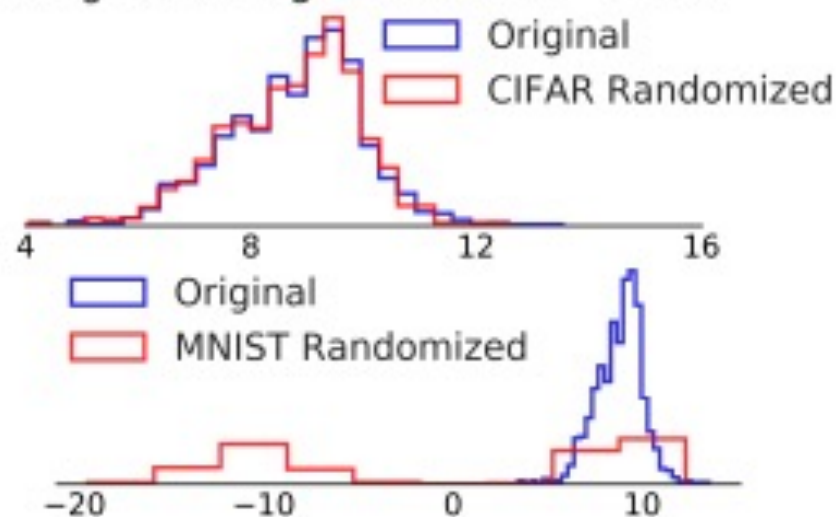
Aditi Raghunathan
Stanford University
aditir@stanford.edu

Prateek Jain
Microsoft Research
prajain@microsoft.com

Praneeth Netrapalli
Microsoft Research
praneeth@microsoft.com



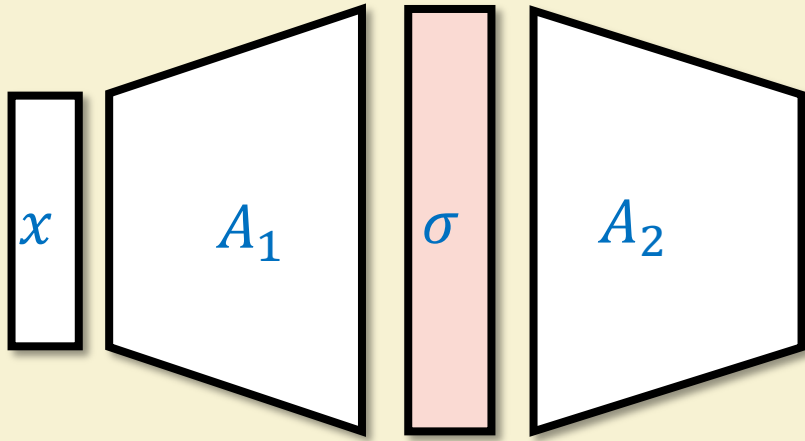
GoogLeNet Logits on MNIST-CIFAR



Model	MNIST-CIFAR: A AUCs		
	Standard	CIFAR10 Randomized	MNIST Randomized
MobileNetV2	1.00 ± 0.00	1.00 ± 0.00	0.53 ± 0.01
GoogLeNet	1.00 ± 0.00	1.00 ± 0.00	0.52 ± 0.02
ResNet50	1.00 ± 0.00	1.00 ± 0.00	0.50 ± 0.01
DenseNet121	1.00 ± 0.00	1.00 ± 0.00	0.53 ± 0.02

Toy Model: Deep Linear Networks

Depth 2 network



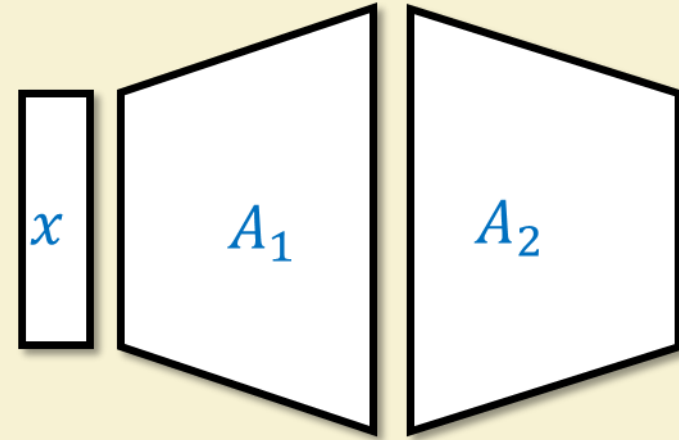
Parameter space: $\mathbb{R}^{d \times h + h \times m}$

$$Bx = A_2 A_1 x:$$

Same **expressiveness** /
functional space

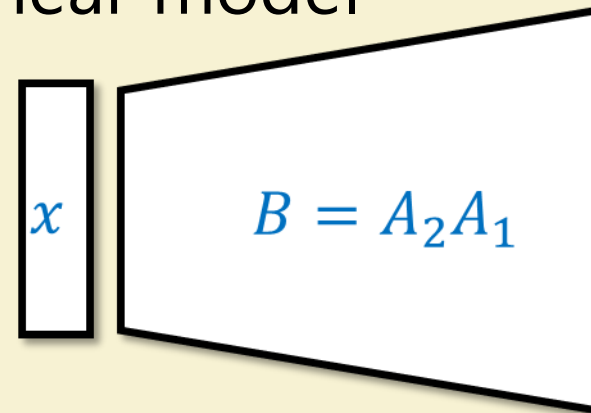
Different **parameter space**

Depth 2 **linear** network



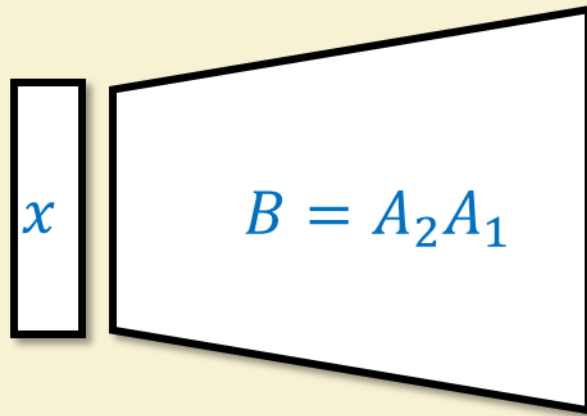
Parameter space: $\mathbb{R}^{d \times h + h \times m}$

Linear model



Parameter space: $\mathbb{R}^{d \times m}$

Linear model



Parameter space: $\mathbb{R}^{d \times m}$

For every loss function \mathcal{L} :

$$\min \mathcal{L}(B)$$

=

BUT

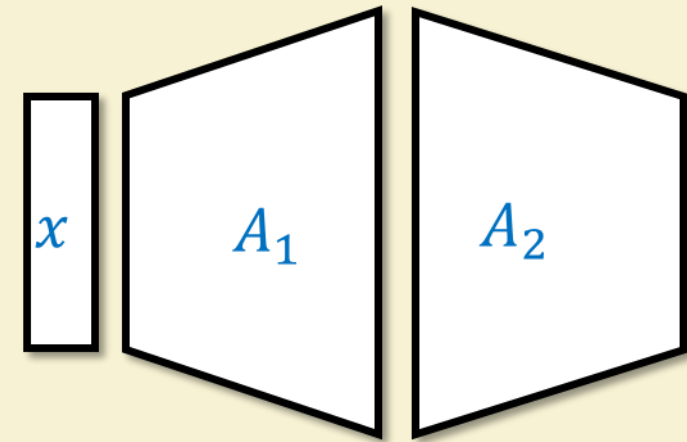
SGD/GD on \uparrow

\neq

Convex function in

$$B \in \mathbb{R}^{d \times m}$$

Depth 2 **linear** network



Parameter space: $\mathbb{R}^{d \times h + h \times m}$

$$\min \mathcal{L}(A_1, A_2)$$

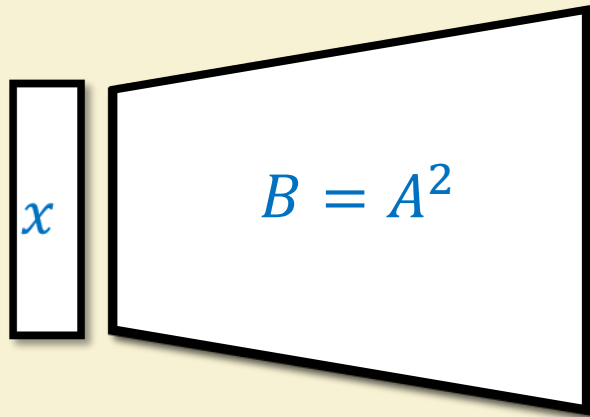
SGD/GD on \uparrow

Non-convex function in

$$(A_1, A_2) \in \mathbb{R}^{d \times h + h \times m}$$

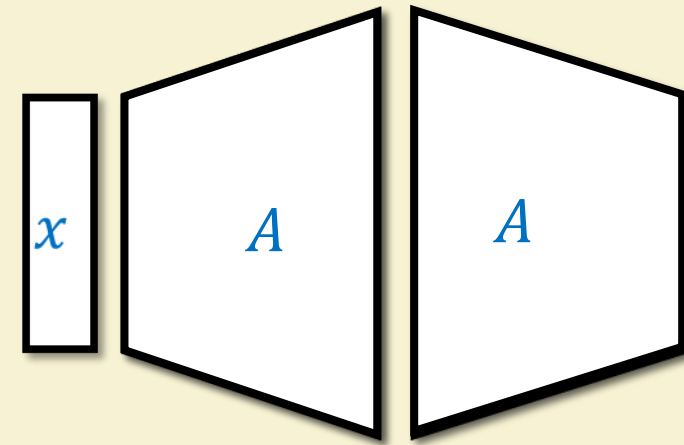
Gradient flow on deep linear nets

Linear model



Parameter space: $\mathbb{R}^{d \times m}$

Depth 2 **linear** network



Parameter space: $\mathbb{R}^{d \times h + h \times m}$

Simplifying assumptions: $A_1 = A_2$ symmetric

$$\Rightarrow B = A^2, A = \sqrt{B}$$

Analyze GD with $\eta \rightarrow 0$ on $\min \tilde{\mathcal{L}}(A)$ where $\tilde{\mathcal{L}}(A) = \mathcal{L}(A^2)$

Gradient flow on deep linear nets

$$\tilde{\mathcal{L}}(A) = \mathcal{L}(A^2)$$
$$B = A^2$$

$$\frac{dA(t)}{dt} = -\tilde{\nabla} \tilde{\mathcal{L}}(A(t))$$

By chain rule $\tilde{\nabla} \tilde{\mathcal{L}}(A) = \nabla \mathcal{L}(A^2) A = A \nabla \mathcal{L}(A^2)$

$\tilde{\nabla} = A \nabla$

GF on linear model:

$$\frac{dB(t)}{dt} = -\nabla \mathcal{L}(B(t))$$

Hence $\frac{dA^2(t)}{dt} = \frac{dA(t)}{dt} \cdot A = -\tilde{\nabla} \cdot A = -A \cdot \nabla \cdot A$

GF on deep linear net $B = A^2$:

$$\frac{dB(t)}{dt} = -A \nabla \mathcal{L}(B(t)) A = -\sqrt{B} \nabla \mathcal{L}(B(t)) \sqrt{B}$$

"The big get bigger"

* dropping 2's throughout

GF on deep linear net $B = A^2$:

$$\frac{dB(t)}{dt} = -A \nabla \mathcal{L}(B(t)) A = -\sqrt{B} \nabla \mathcal{L}(B(t)) \sqrt{B}$$

Generally GF on deep linear net B evolves* by $\frac{dB(t)}{dt} = -\psi_{B(t)}(\nabla \mathcal{L}(B(t)))$

$$\psi_B(\nabla) =^* \sum B^\alpha \nabla B^{1-\alpha}$$

Gradient flow on a **Riemannian Manifold**

* **not** equivalent to $\min \mathcal{L}(B) + \lambda R(B)$

Saxe, McClelland, Ganguli 2013

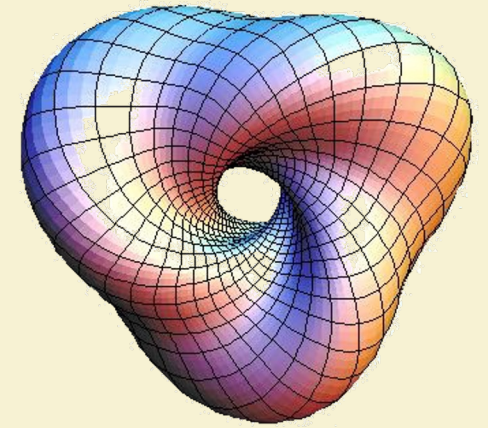
Arora, Cohen, Hazan, 2018

Bah, Rauhut, Terstiege, Westdickenberg, 2019

Riemannian Manifolds

External description: A smooth subset $\mathcal{M} \subseteq \mathbb{R}^N$

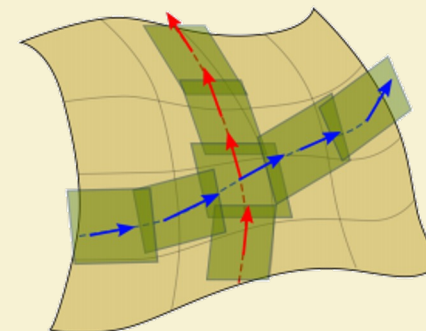
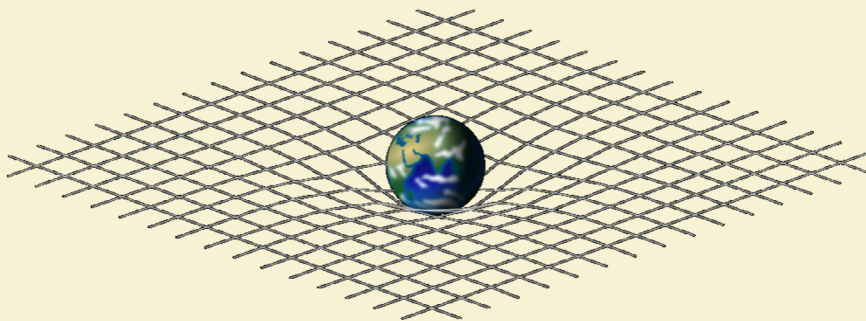
Intrinsic description: Set \mathcal{M} with "local geometry" at each $x \in \mathcal{M}$



For every $x \in \mathcal{M}$, **tangent space** T_x - set of directions we can move in

(Gradient of $f(x)$: shortest direction from x to increase f)

local inner product on T_x - defined via PSD matrix M_x on T_x

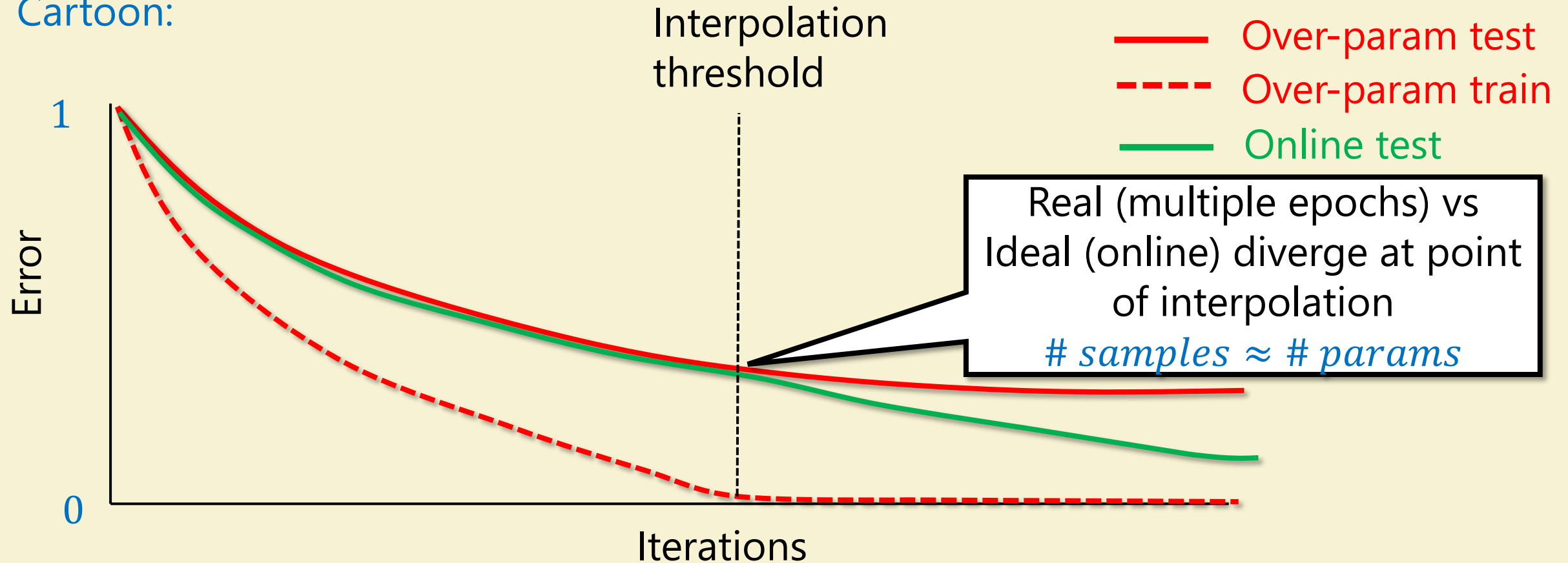


Does over-parameterization matter?

Deep Bootstrap [Nakkiran, Neyshabur, Sedghie ICLR '21]

Compare 100 epochs on 50K samples w/ 1 epoch on 5M samples.

Cartoon:



Does over-parameterization matter?

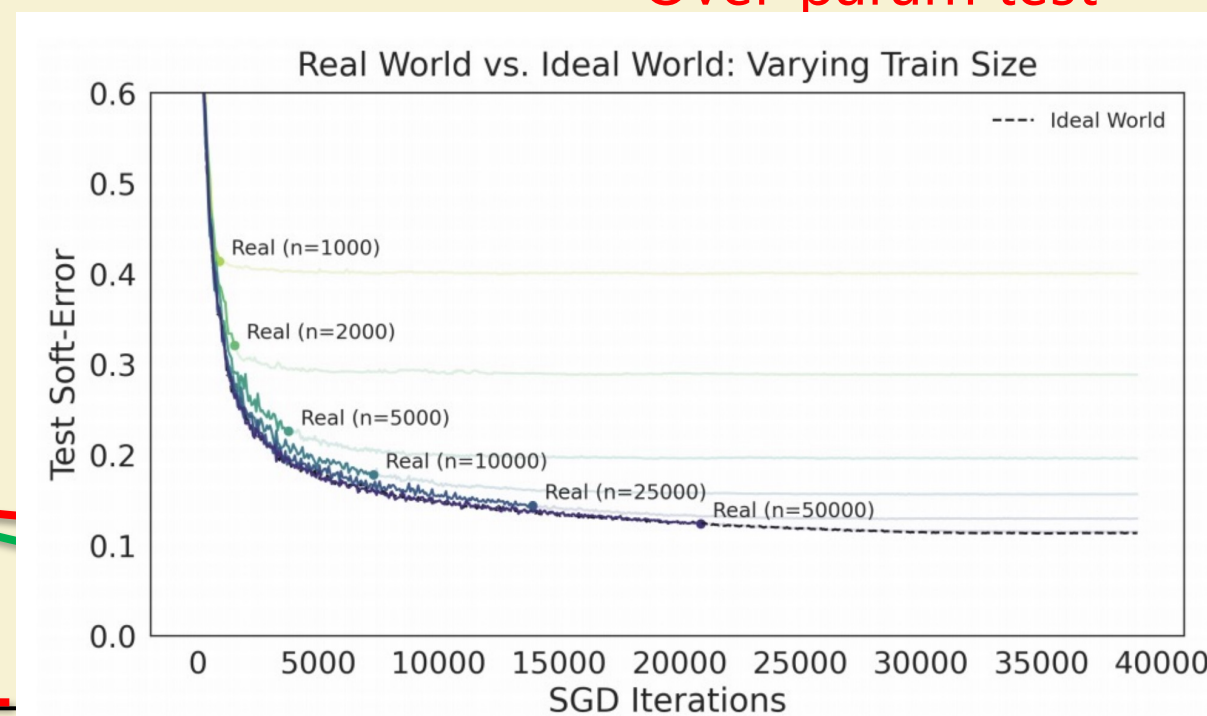
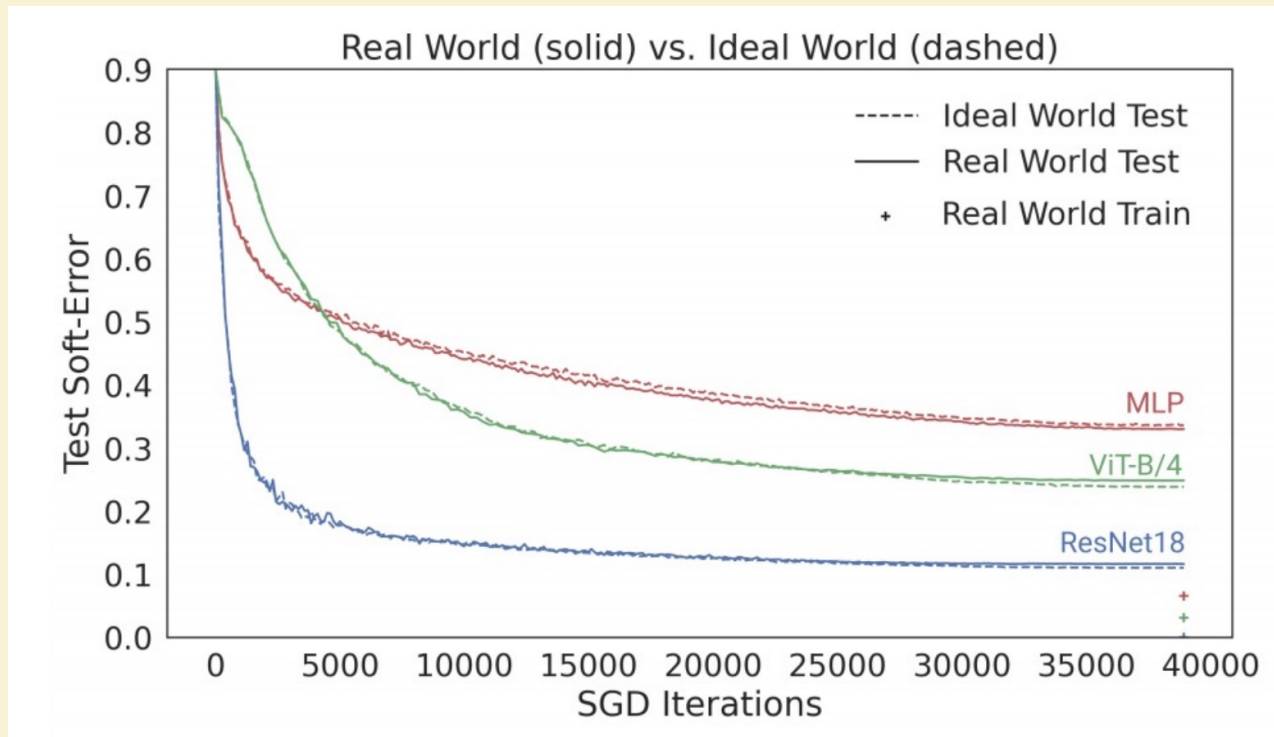
Deep Bootstrap [Nakkiran, Neyshabur, Sedghie ICLR '21]

Compare 100 epochs on 50K samples w/ 1 epoch on 5M samples.

Cartoon:

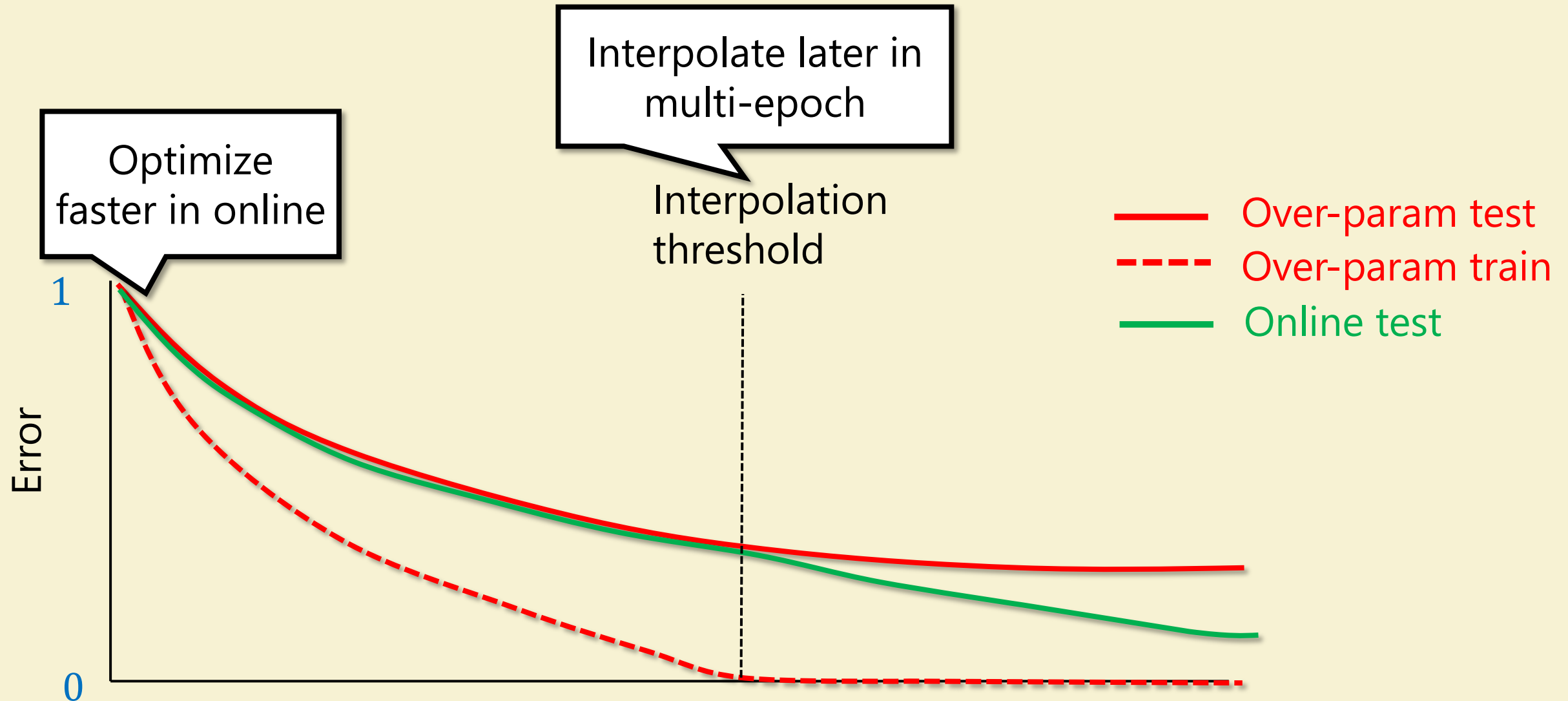
Interpolation

— Over-param test



Iterations

Conclusions: To get better performance



Even in online setting, no reason to stop before $\# \text{ steps} \approx \text{model capacity}$

Toy model: Kernel Methods

Ultra wide NNs
 \approx Kernels

Linear Regression: Input: $x_1, y_1, \dots, x_n, y_n \in \mathbb{R}^{d+1}$

Goal: Find $w \in \mathbb{R}^d$ minimizing $\|Xw - y\|^2$

Be able to compute $x_* \mapsto \langle x, w \rangle$

"Kernel Trick": Suppose $n \ll d$, we want $w = \sum_i \alpha_i x_i$

$$\alpha = \arg \min \|XX^T \alpha - y\|^2$$

"Look Ma no d "
 n equations in
 n variables

Given new point x^* , $\langle w, x^* \rangle = x^* X w = \sum \alpha_i \langle x^*, x_i \rangle$

"black box" for $\langle x, x_i \rangle \Rightarrow$ compute in $\tilde{O}(n)$ instead of $\tilde{O}(d)$

Need: Implicit representation of $x \in \mathbb{R}^d$ + alg for dot products

Deep Bootstrap, Simplicity bias and Kernel / LR

$$\hat{L}(\theta) = L_{\text{signal}}(\theta) + L_{\text{noise}}(\theta)$$

Online setting: $\mathbb{E} \nabla L_{\text{noise}} = 0$

Training
loss

1

Error

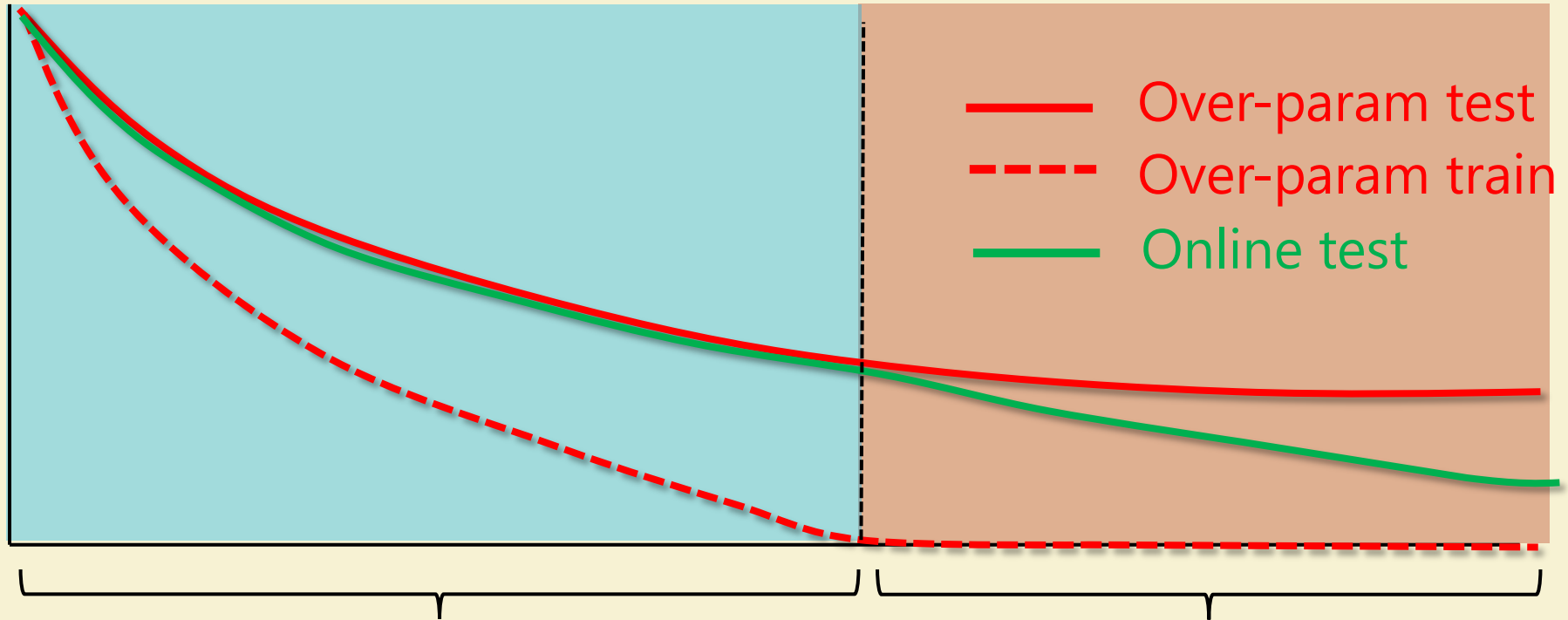
0

Interpolation
threshold

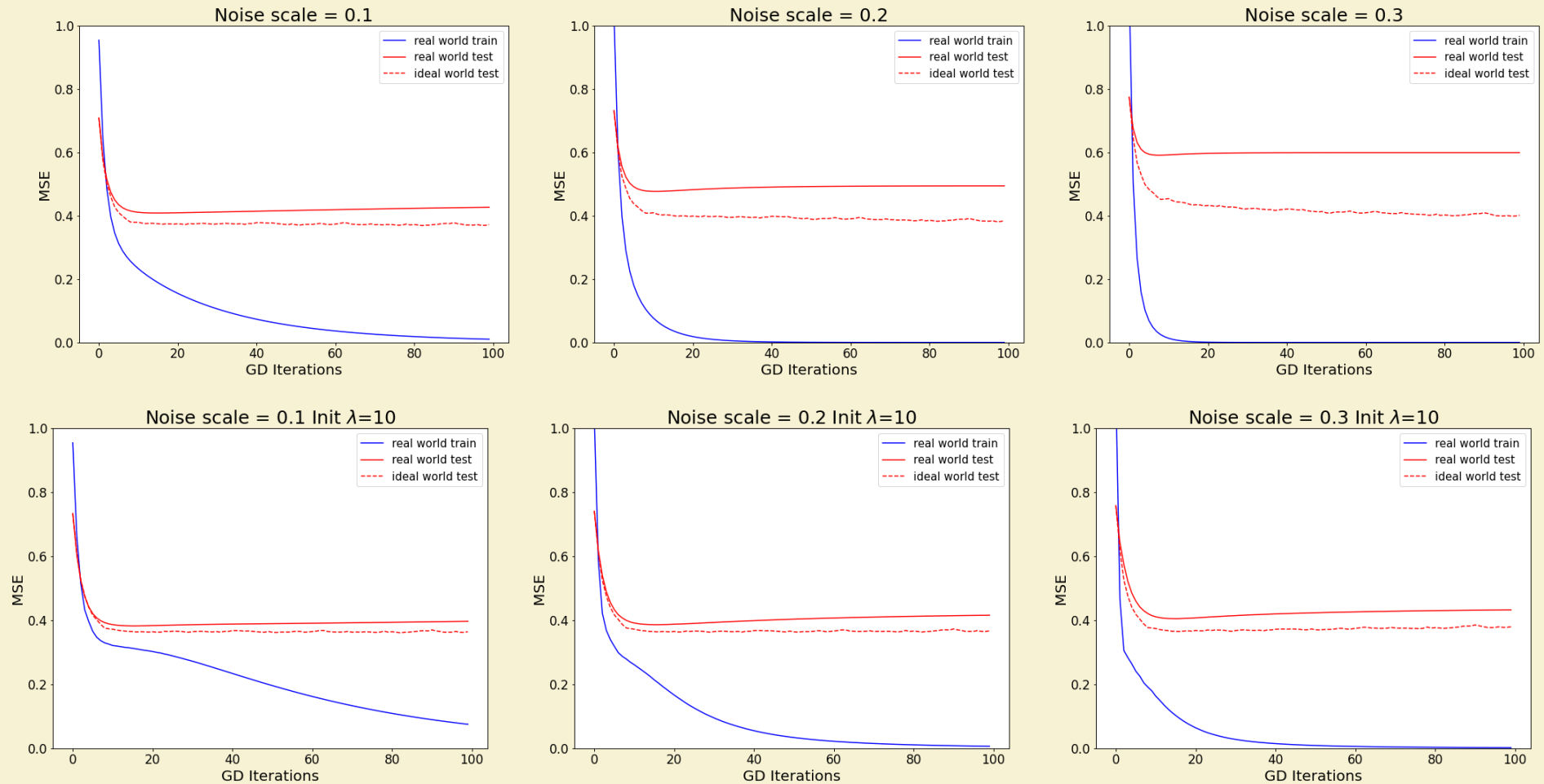
— Over-param test
- - - Over-param train
— Online test

Generalization -
fitting signal

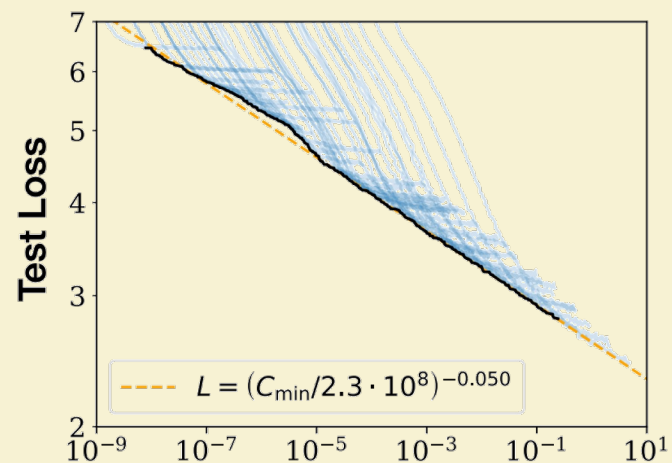
Memorization -
fitting noise



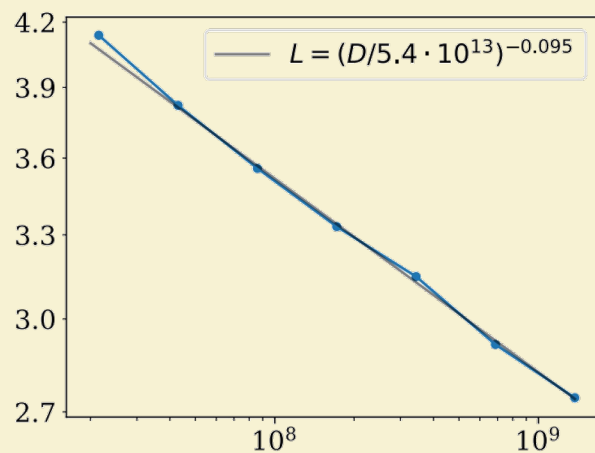
Simplicity bias and deep bootstrap



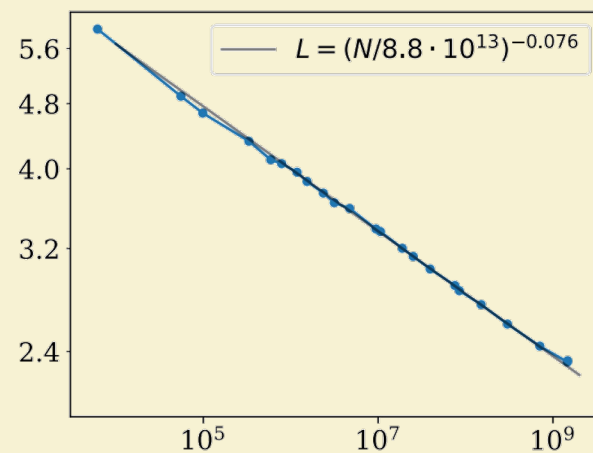
"Scaling Laws"



Compute
PF-days, non-embedding

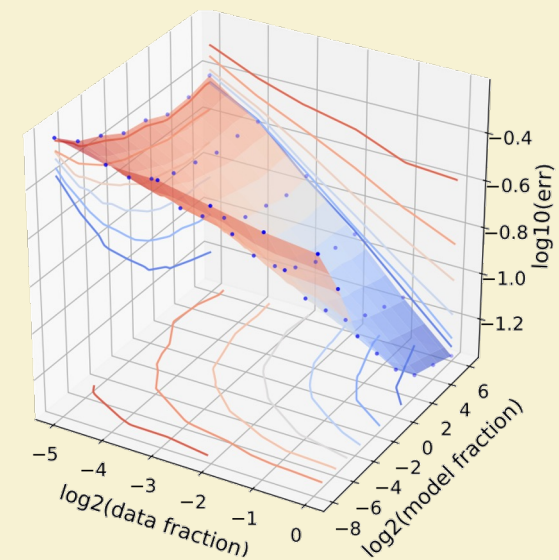


Dataset Size
tokens



Parameters
non-embedding

Kaplan et al 20



(b) CIFAR10 error (top1) landscape.

Rosenfeld 19

Strong version

Model
size

Data*
size

$$L(N, D)$$

“Conjecture”: For all “reasonable” architecture and tasks

Optimum is
 $D \propto N^{\alpha/\beta}$

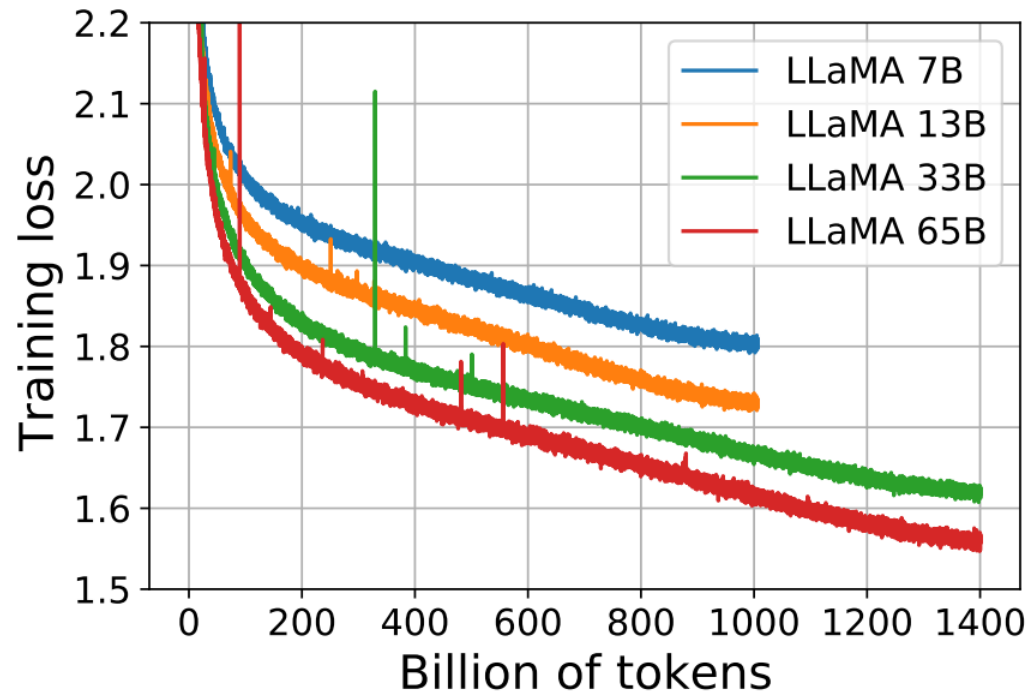
$$L(N, D) \approx \frac{A}{N^\alpha} + \frac{B}{D^\beta} + E$$

α, β independent of
architecture /
algorithm

Kaplan et al: $\alpha \approx 0.08$, $\beta \approx 0.1$, $D \propto N^{0.74}$

Chinchilla: $\alpha, \beta \approx 0.3$ ($\alpha \approx 0.34$, $\beta \approx 0.28$) , $D \propto N$

Are scaling laws broken?



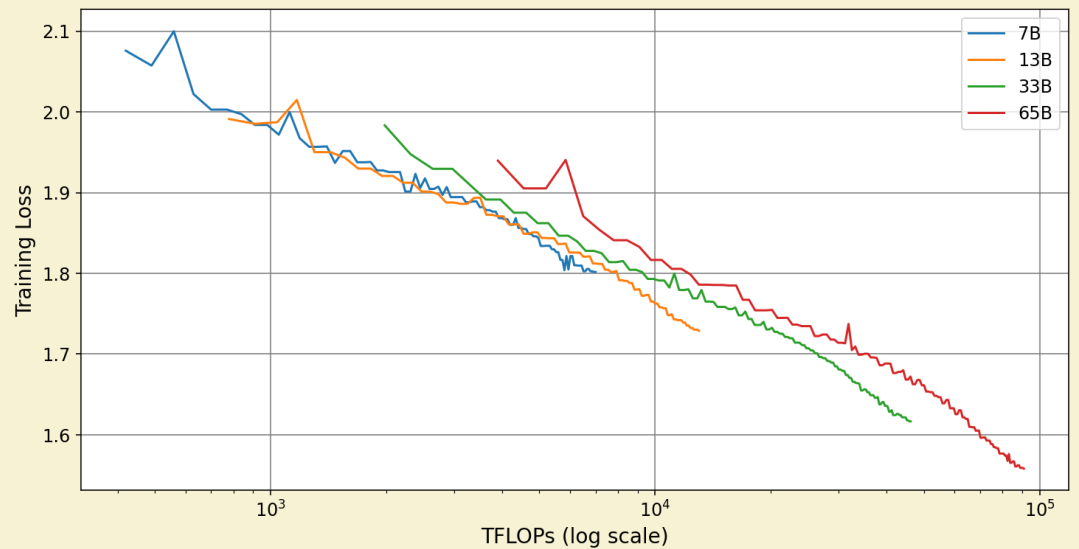
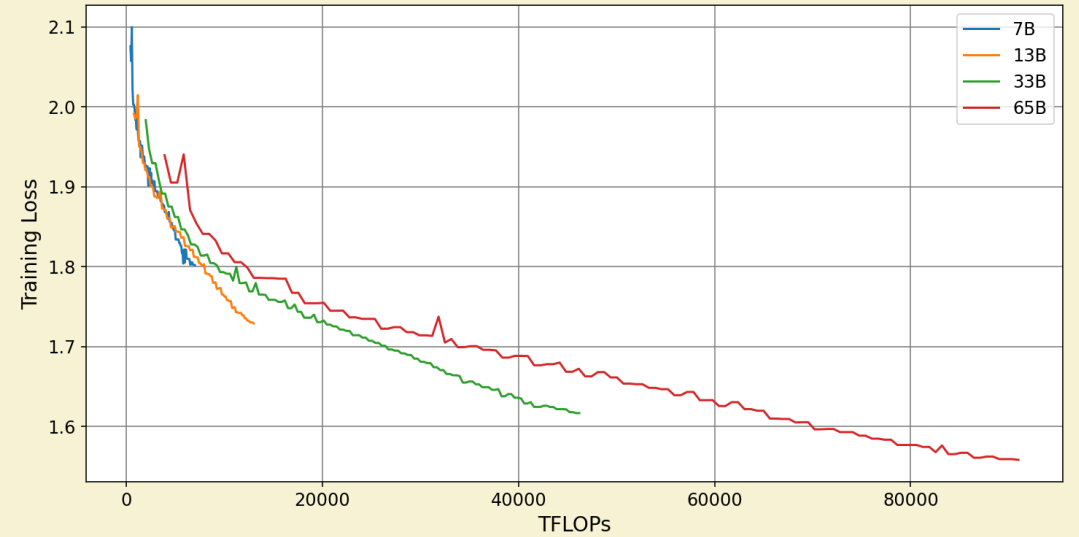
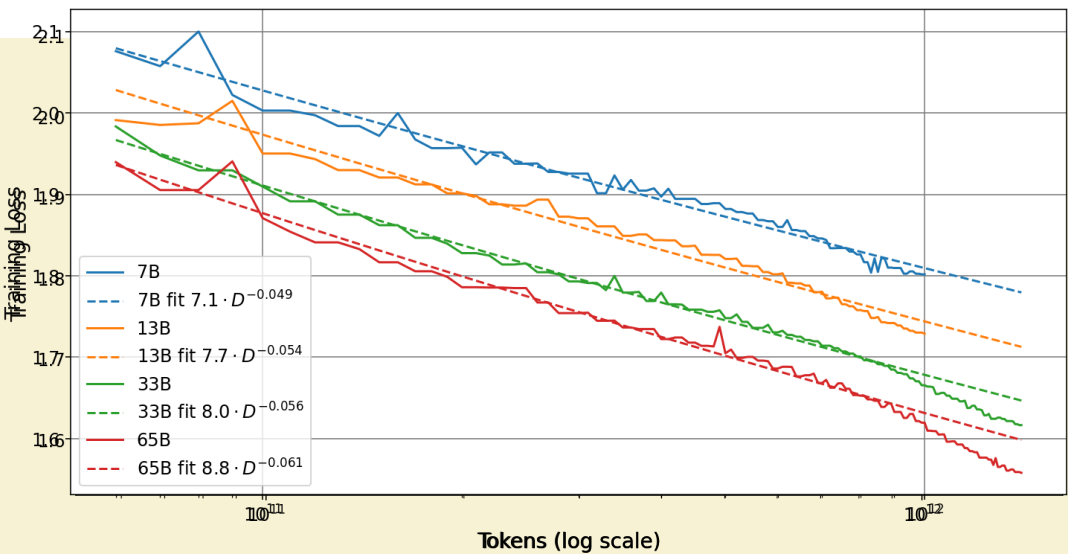
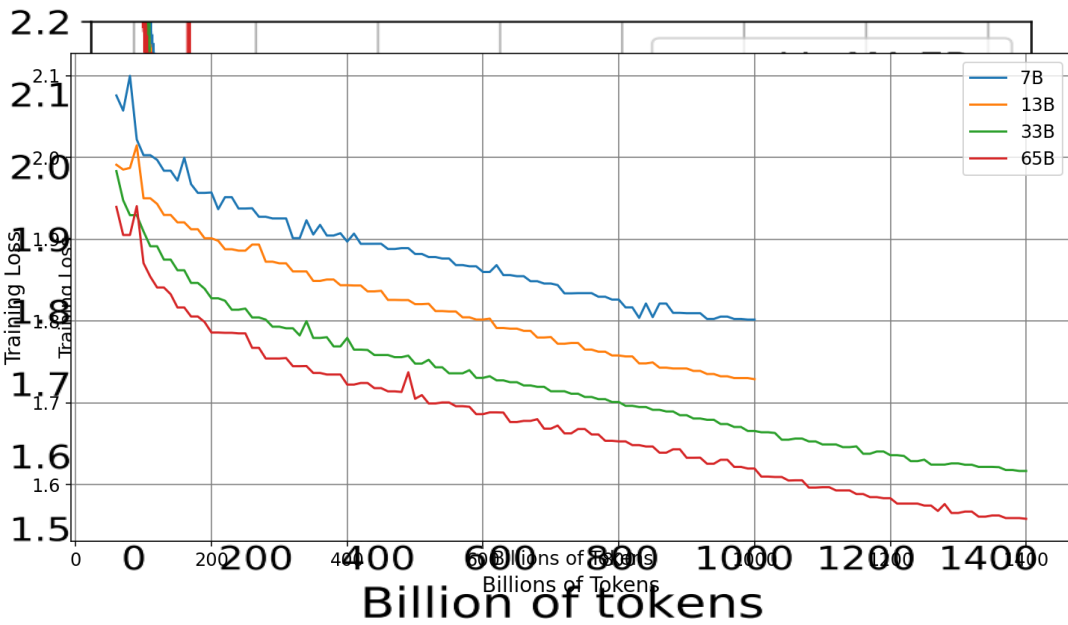
LLaMA: Open and Efficient Foundation Language Models

Hugo Touvron,* Thibaut Lavril,* Gautier Izacard,* Xavier Martinet
Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal
Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin
Edouard Grave,* Guillaume Lample*



LLaMA

As function of compute



Critical Truths About Power Laws

Most reported power laws lack statistical support and mechanistic backing.

Michael P. H. Stumpf¹ and Mason A. Porter²

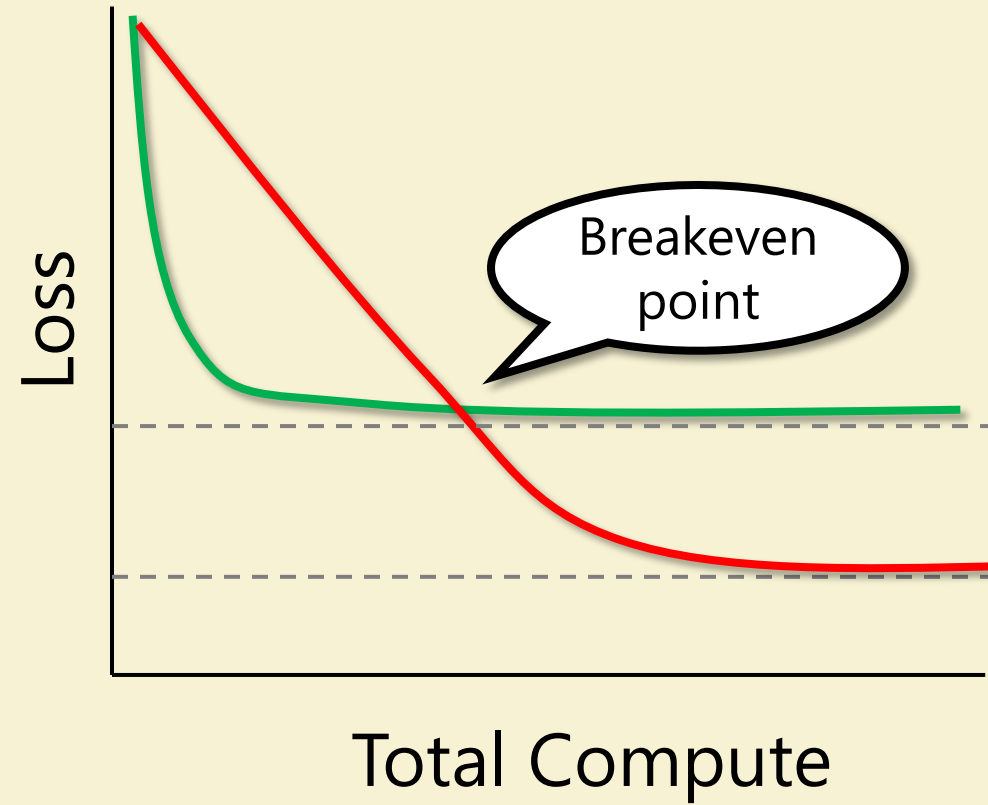
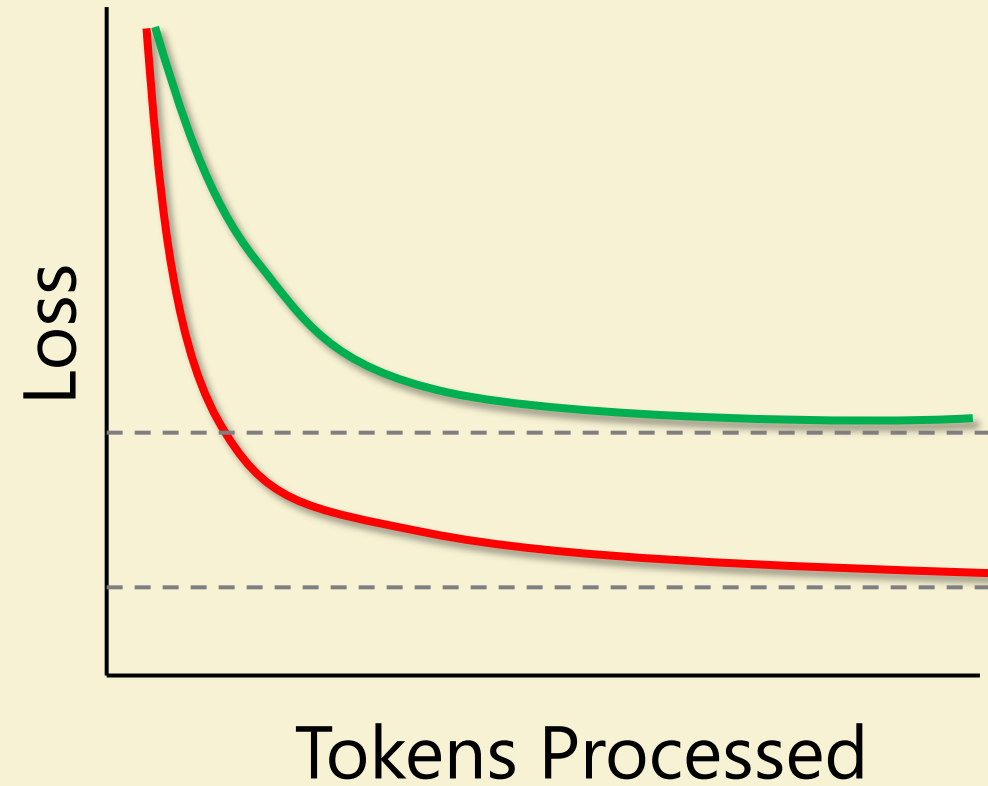
POWER-LAW DISTRIBUTIONS IN EMPIRICAL DATA

AARON CLAUSET^{*}, COSMA ROHILLA SHALIZI[†], AND M. E. J. NEWMAN[‡]

Abstract. Power-law distributions occur in many situations of scientific interest and have significant consequences for our understanding of natural and man-made phenomena. Unfortunately, the detection and characterization of power laws is complicated by the large fluctuations that occur in the tail of the distribution—the part of the distribution representing large but rare events—and by the difficulty of identifying the range over which power-law behavior holds. Commonly used methods for analyzing power-law data, such as least-squares fitting, can produce substantially inaccurate estimates of parameters for power-law distributions, and even in cases where such methods return accurate answers they are still unsatisfactory because they give no indication of whether the data obey a power law at all. Here we present a principled statistical framework for discerning and quantifying power-law behavior in empirical data. Our approach combines maximum-likelihood fitting methods with goodness-of-fit tests based on the Kolmogorov-Smirnov statistic and likelihood ratios. We evaluate the effectiveness of the approach with tests on synthetic data and give critical comparisons to previous approaches. We also apply the proposed methods to twenty-four real-world data sets from a range of different disciplines, each of which has been conjectured to follow a power-law distribution. In some cases we find these conjectures to be consistent with the data while in others the power law is ruled out.

What seems true

— Large Model
— Small Model



Toy model: k -Nearest Neighbor

Assume that “under the hood”, training deep net on data $x_1, y_1 \dots x_N, y_N$ corresponds to:

- Learning somehow a d -dimensional manifold and embedding φ of x_i 's into this manifold.
- Model's output on new x obtained by combining y 's for k x_i 's closest to x in this manifold (Interpolating classifier: $k = 1$)

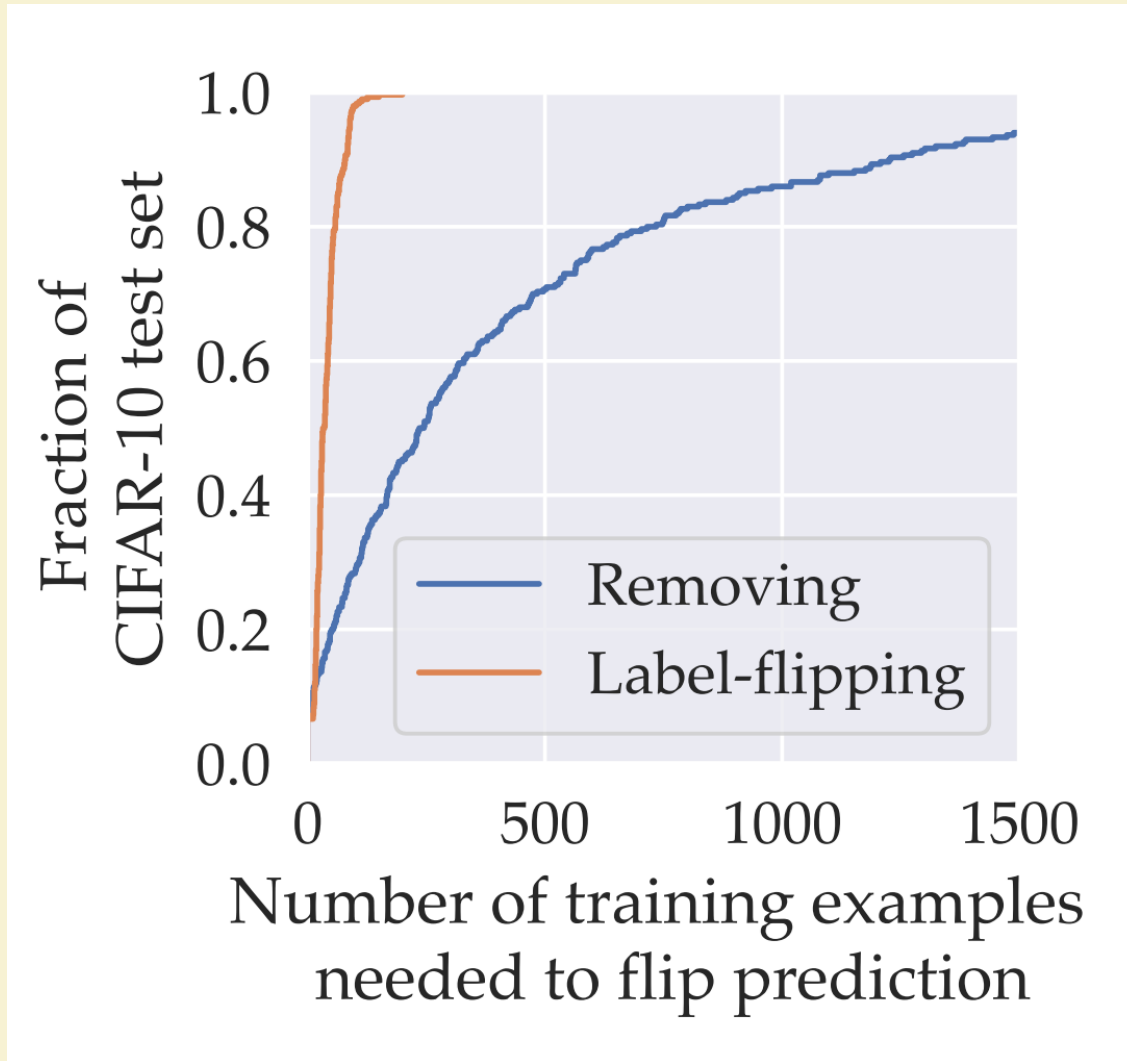
Not mechanistic, but can still make qualitative & quantitative predictions

Predictions of k -NN model

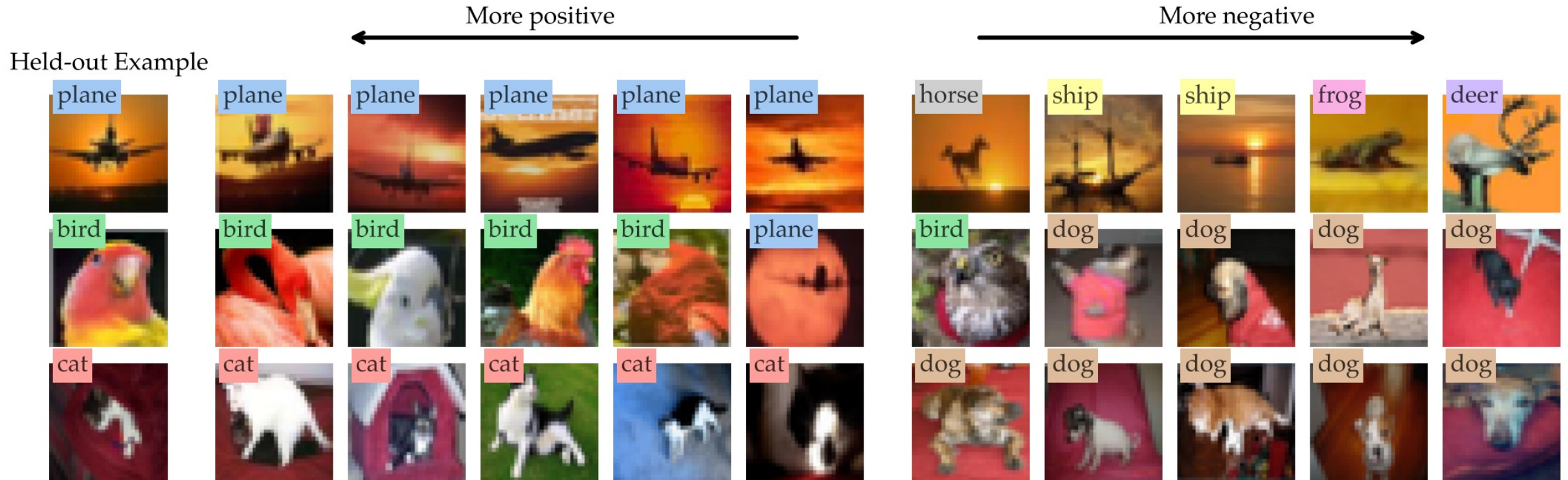
For every x , there would be small set $S(x)$ of points in training set that greatly influence $f(x)$

Whether x influences x' induces a meaningful distance

Datamodels [Ilyas, Park, Engstrom, Leclerc, Madry '22]

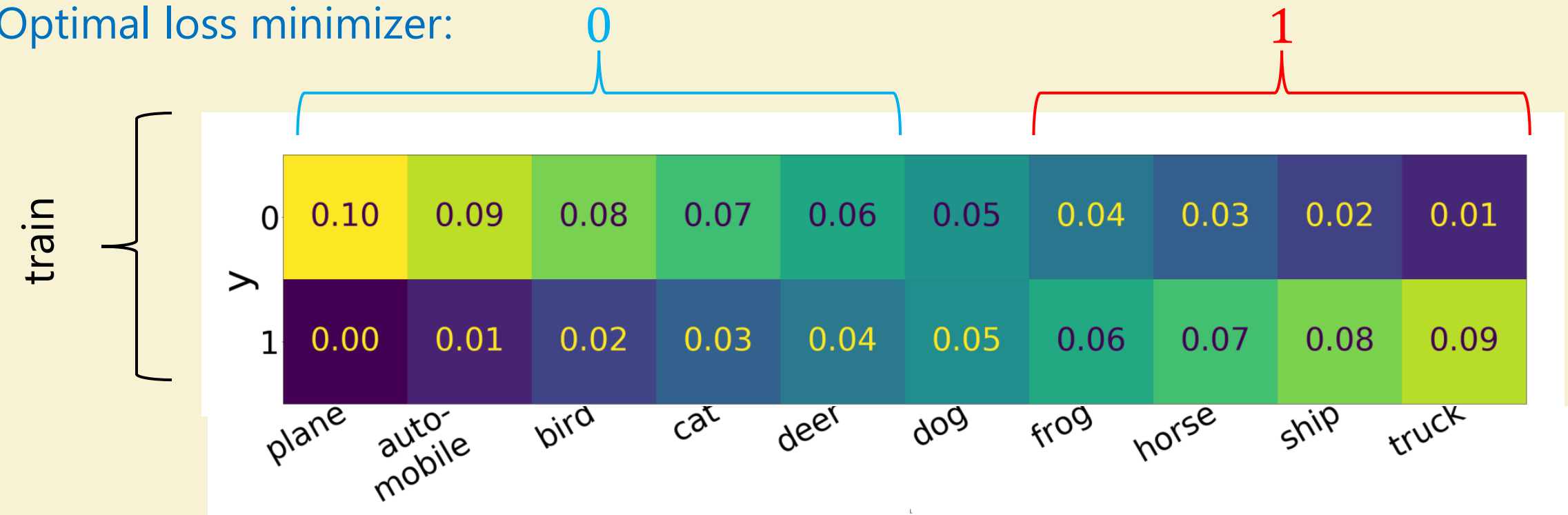


Datamodels [Ilyas, Park, Engstrom, Leclerc, Madry '22]

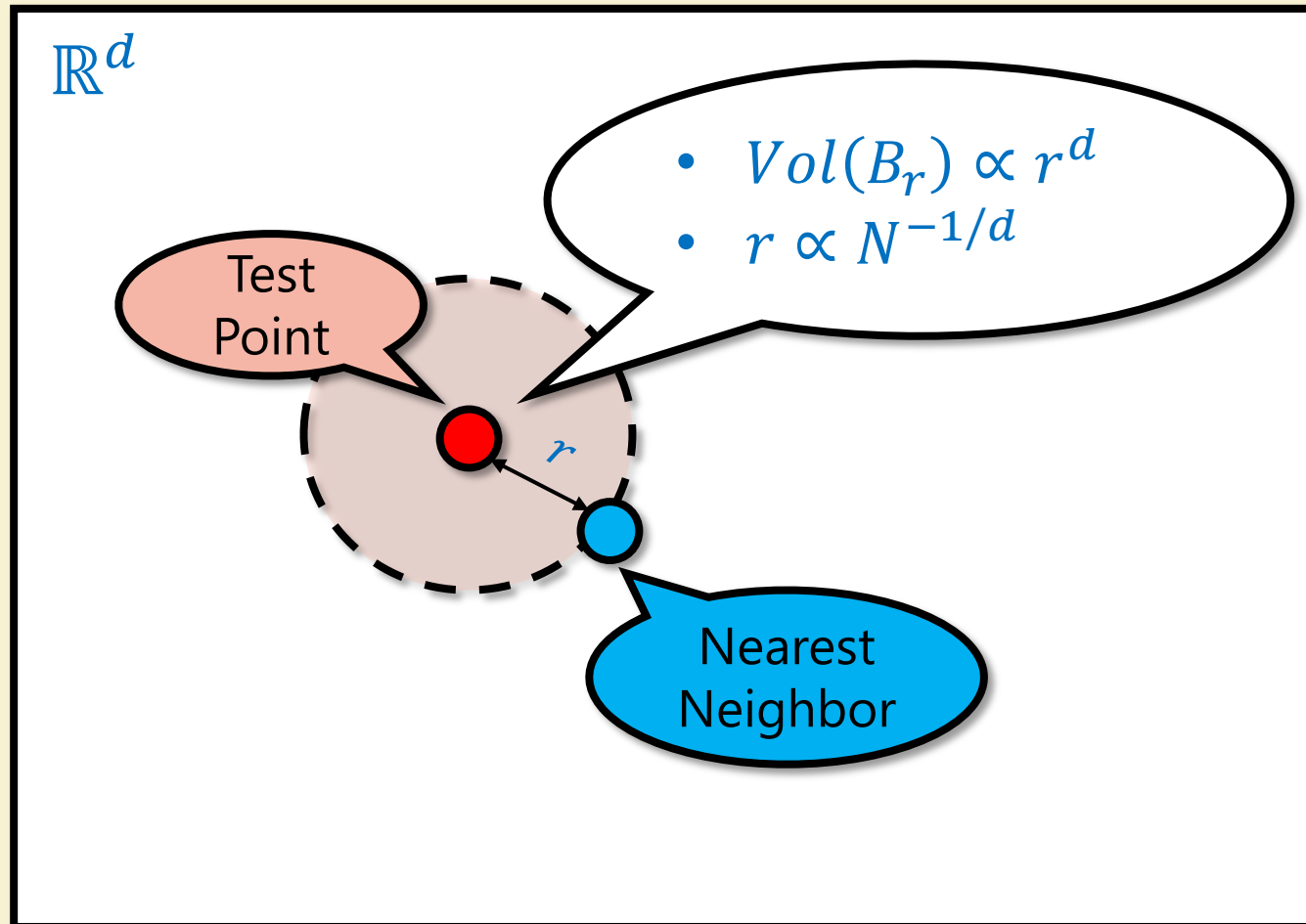


Distributional Generalization [Nakkiran-Bansal'20]

Optimal loss minimizer:



Scaling laws from nearest neighbor



Edge of Stability

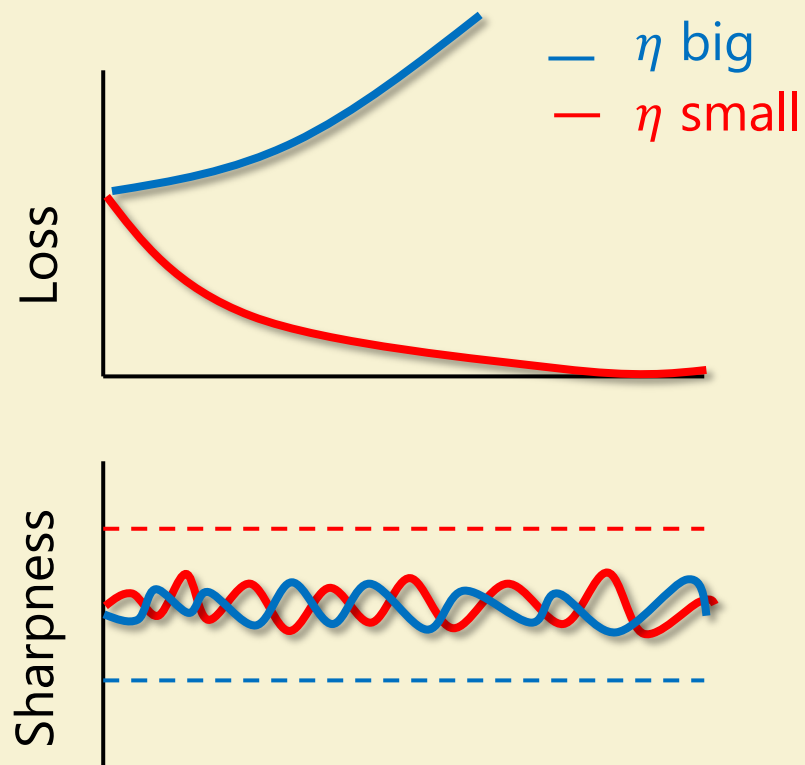
$$\hat{w}_t := w_t - w_\infty$$

Sharpness

GD: $\hat{w}_{t+1} = \left(I - \frac{\eta}{2}H\right) \hat{w}_t$

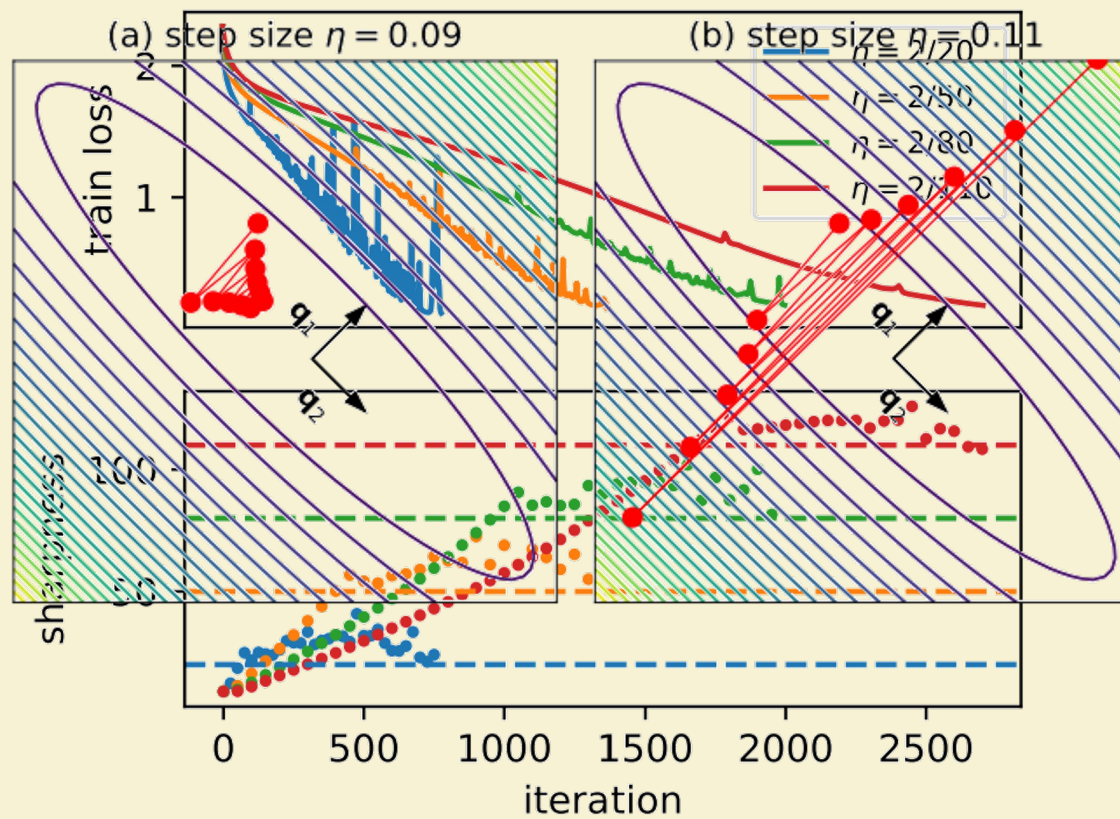
Convergence requires $\lambda_{\max}(H) < \frac{2}{\eta}$

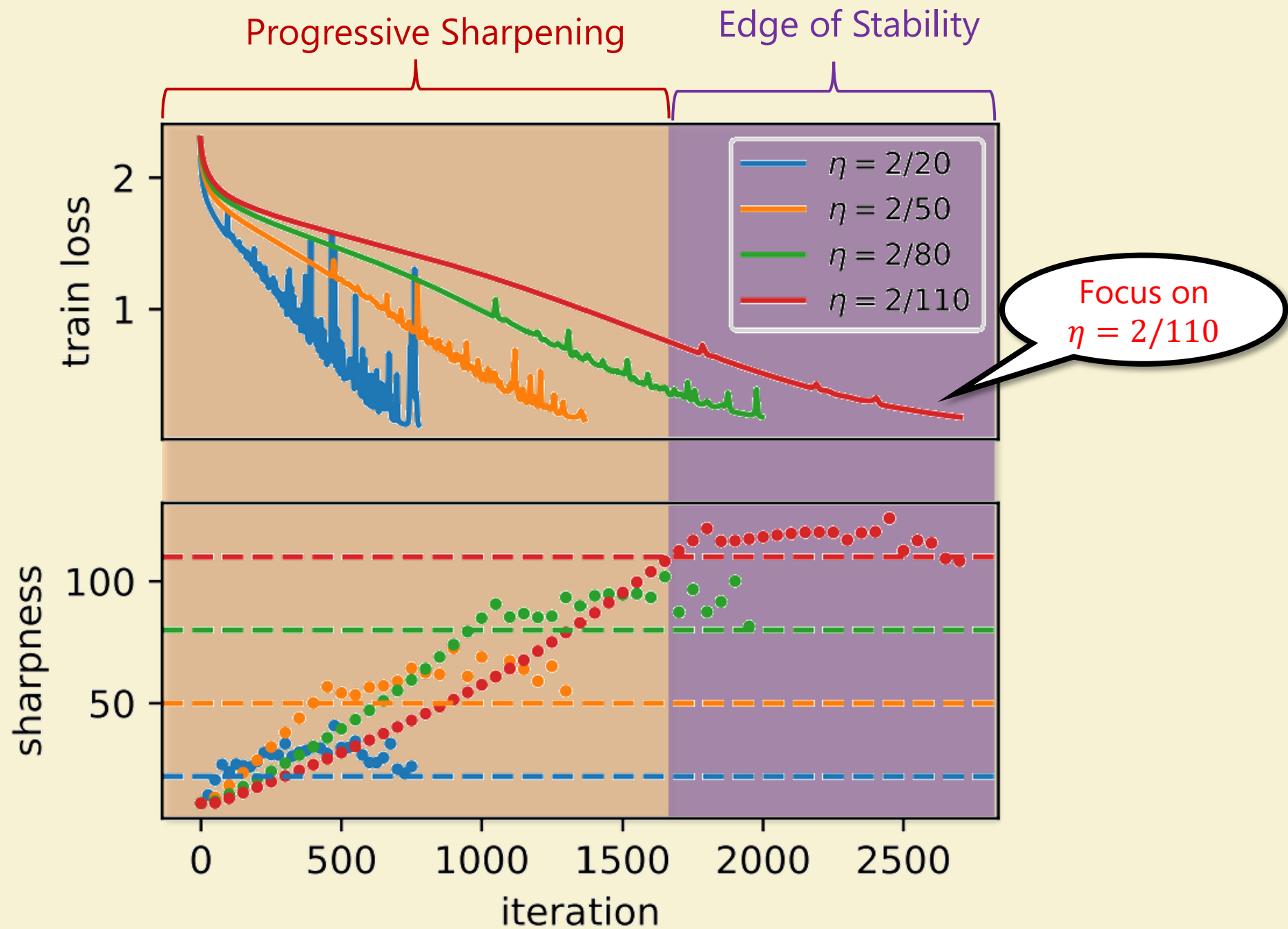
Expect:



Actual [Cohen et al 2021]

Fully-connected net on CIFAR-10 5k subset

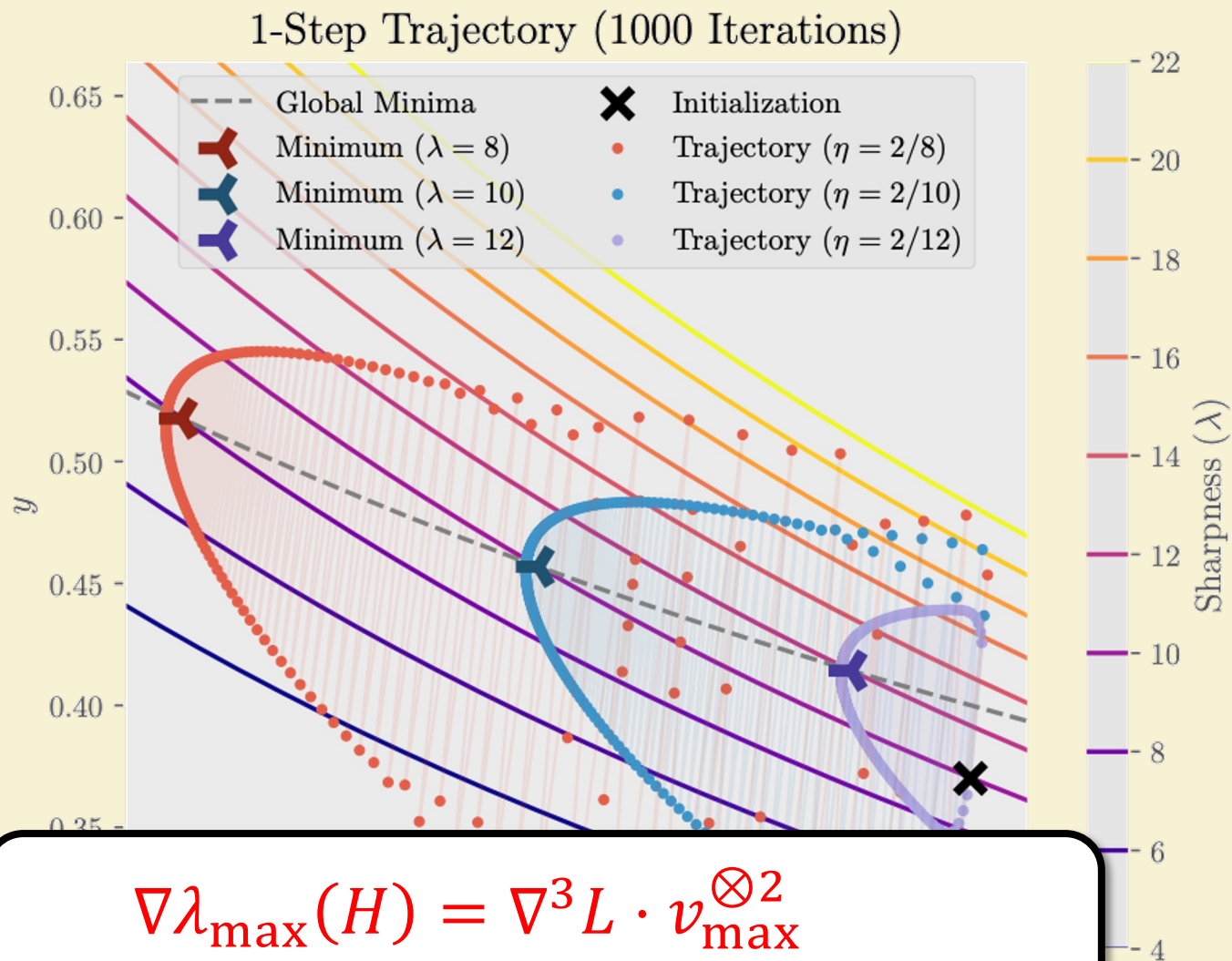
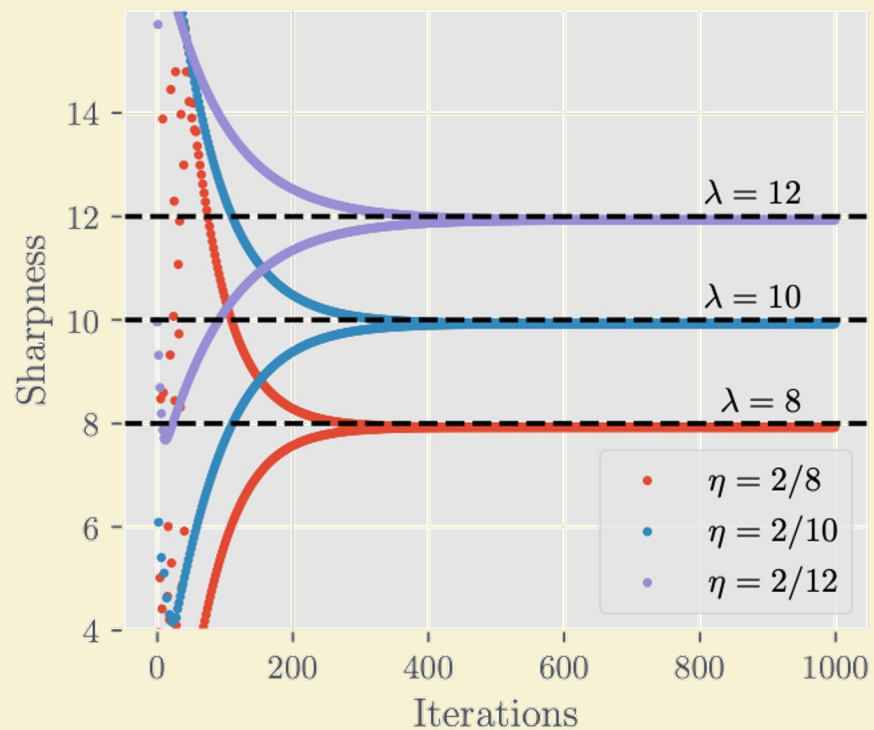
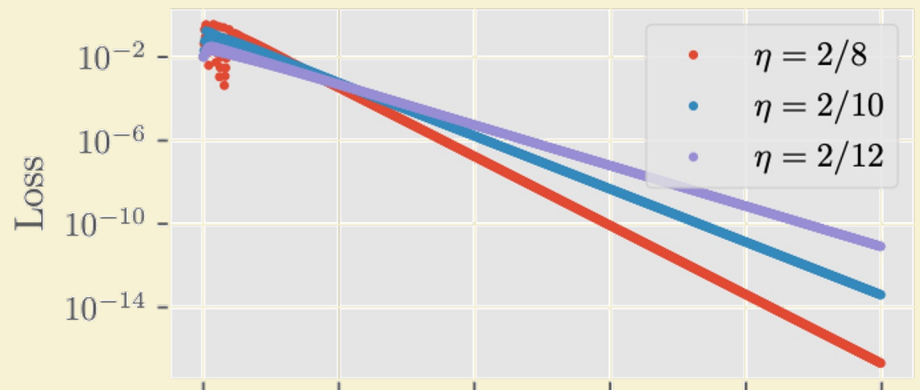




Explaining edge of stability

$$L(x, y) = \frac{1}{4}(1 - x^2 y^2)^2$$

[Zhu et al 2023]



$$\nabla \lambda_{\max}(H) = \nabla^3 L \cdot v_{\max}^{\otimes 2}$$

[Damian et al 22]

