CS 760: Machine Learning
Science of Deep Learning

University of Wisconsin-Madison

Last lecture

SVMs:

- strong learning-theoretic motivation (maximize the margin)

$$generalization\ error \leq \frac{2}{\rho\sqrt{m}} + \sqrt{\frac{1}{2m}\log\frac{1}{\delta}}$$

- well-understood optimization problem

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 \quad \text{subject to} \quad y_i(w^\top x_i + b) \geq 1 \ \forall\, i$$

Can we hope to get the same for deep learning?

Outline

- Limitations of traditional learning theory
  - fitting random labels, double descent
- Limitations of optimization theory
  - edge of stability
- Towards a predictive science of deep learning
  - the central flow, scaling laws

Outline

- Limitations of traditional learning theory
  - fitting random labels, double descent
- Limitations of optimization theory
  - edge of stability
- Towards a predictive science of deep learning
  - the central flow, scaling laws

What does traditional learning theory say?

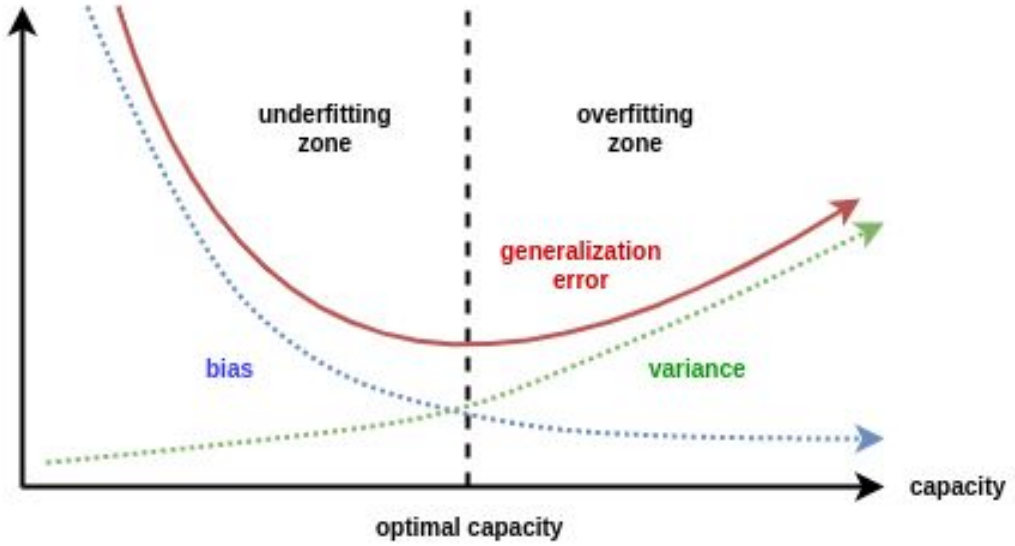
capacity-dependent bounds:

$$generalization\ error = O\left(\sqrt{\frac{capacity(H)}{m}\log\frac{1}{\delta}}\right)$$

- often shown simultaneously $\forall\ h \in H$ (uniform convergence)
- $capacity(H)$ is higher the more easily H overfits to the data

bias-variance tradeoff:

- too much capacity leads to overfitting the data
- need to regularize



Meanwhile, in the real world

The early deep learning revolution (2012-2017) witnessed big models working well on small datasets

model	parameter count	CIFAR-10 test accuracy
ResNet-110	1.7 million	93.57%
WideResNet-28-10	36.5 million	96.2%

- both networks can get zero **training** error
- CIFAR-10 only has 50K training examples!

The learning theory cope

- "Don't plug in numbers into generalization bounds, just trust the guidance of the capacity measures."

- "SGD implicitly searches over a much smaller subset of H; if we can identify that we'll get something meaningful."
  - implicit regularization
  - algorithmic stability

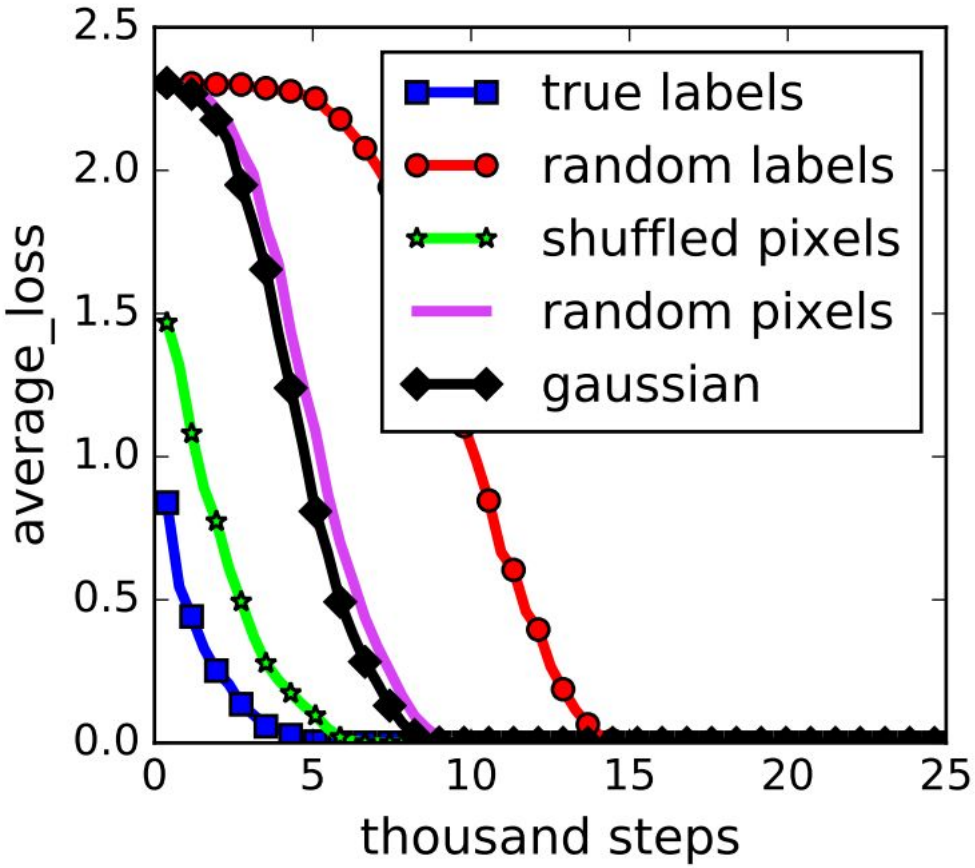
Understanding deep learning requires **rethinking generalization**

In 2017, Zhang et al. report that CNNs easily overfit

1. correctly labeled CIFAR-10 images
2. **randomly** labeled CIFAR-10 images
3. **randomly** labeled Gaussian **noise**

Quiz: what are the generalization errors in each case (roughly)?

1. small (a few %)
2. 90%
3. 90%



What does this mean for generalization?

"Don't plug in numbers into generalization bounds, just trust the guidance of the capacity measures."

Classical capacity measures are defined by the ability of hypotheses in H to fit

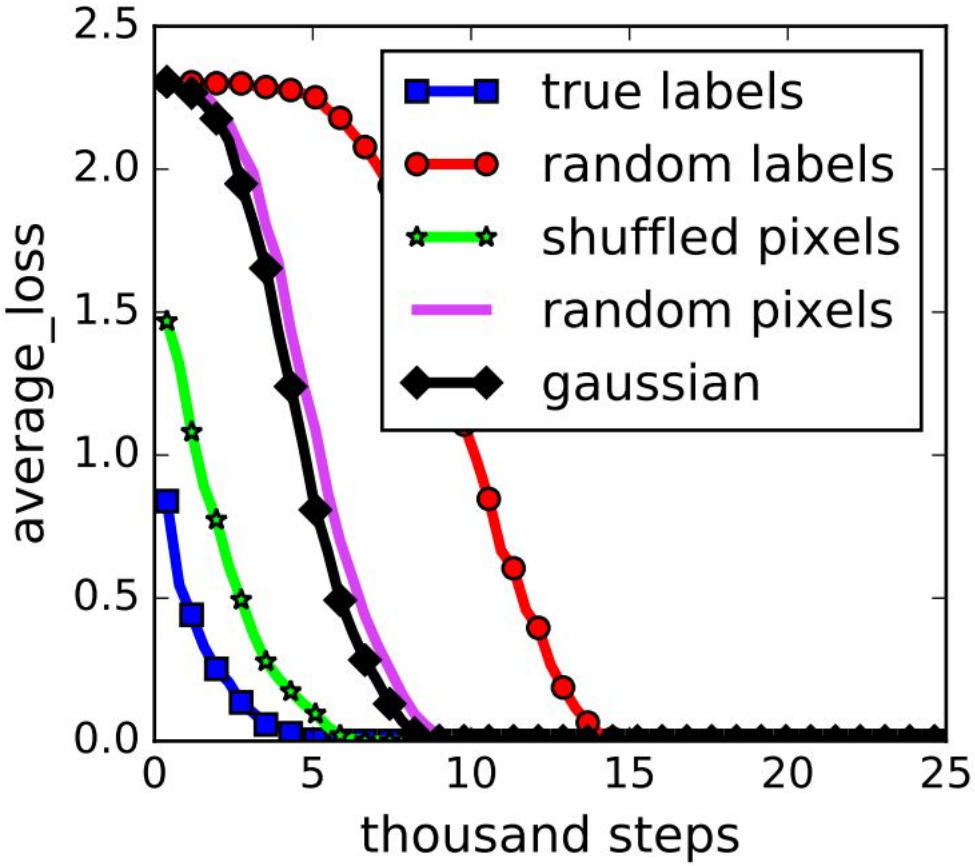
- arbitrary labels (VC-dimension)
- random labels (Rademacher complexity)

But CNNs can do so but still generalize well in practice!

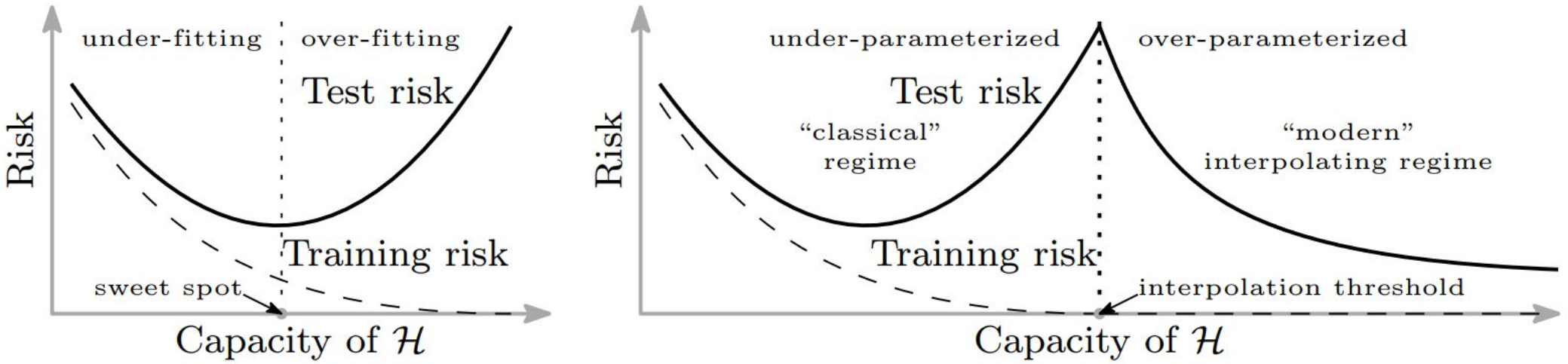
What does this mean for generalization?

"SGD implicitly searches over a much smaller subset of H; if we can identify that we'll get something meaningful."

- SGD (fairly) easily find CNNs that fit random labels
- regularization like weight-decay does not act as a capacity constraint but as a way to improve optimization



Doesn’t this go against the bias-variance tradeoff?

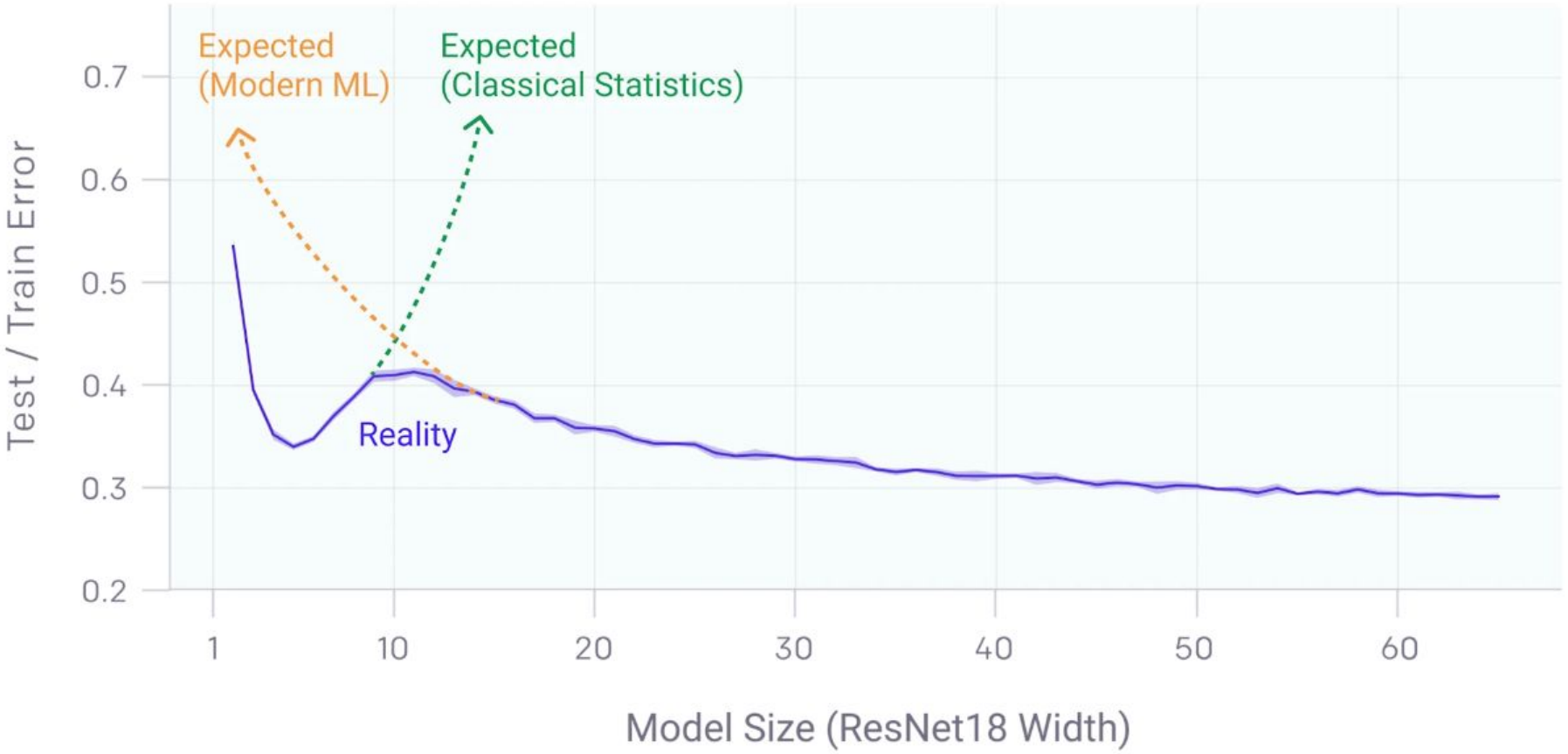


In 2019, Belkin et al. identify double descent:

- generalization improves again after an interpolation threshold
- identified in kernel methods, random forests, and simple MLPs
- “benign overfitting”

Deep double descent

Phenomenon observed in deep CNNs by OpenAI



So what now?

Traditional learning theory

- does not explain generalization in modern deep nets
- is not sufficiently predictive to guide the development of neural network architectures or learning algorithms

Perhaps we can at least use optimization theory to develop better training algorithms?

Outline

- Limitations of traditional learning theory
  - fitting random labels, double descent
- Limitations of optimization theory
  - edge of stability
- Towards a predictive science of deep learning
  - the central flow, scaling laws

Recall: What do optimization guarantees look like?

If f is convex and has L-Lipschitz gradients, and if we run gradient descent with step-size $\eta \leq 1/L$ starting at x_0, then the Tth iterate has suboptimality

$$f(x_T) - \min_x f(x^*) \leq \frac{||x_0 - x^*||_2^2}{2T\eta}$$

In non-convex settings (deep nets) we show convergence to a stationary point. Many algorithms have such guarantees.

Can we use optimization theory to design better optimization algorithms?

People certainly try: one example (Google's LAMB algorithm)

- assume objective has L_i-Lipschitz gradients w.r.t. layer i
- LAMB **adapts to the per-layer smoothness**

$$sub-optimality = O\left(\frac{\frac{1}{h}\sum_{i=1}^{h} L_i}{T}\right)$$

- compare to the gradient descent guarantee:

$$sub-optimality = O\left(\frac{\max_i L_i}{T}\right)$$

On the other hand: by far the most popular optimizer (Adam) originally had an incorrect convergence proof

So does classical optimization theory explain the convergence of gradient descent for deep nets?

Recall that for L-smooth f we had to use step size $\eta \leq 1/L$

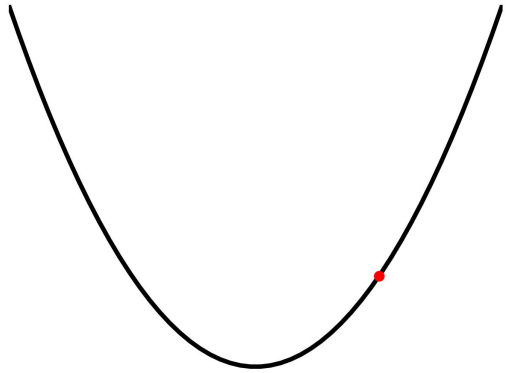
For quadratics, we can get away with $\eta \leq \mathbf{2}/L$

Why can't we go higher?

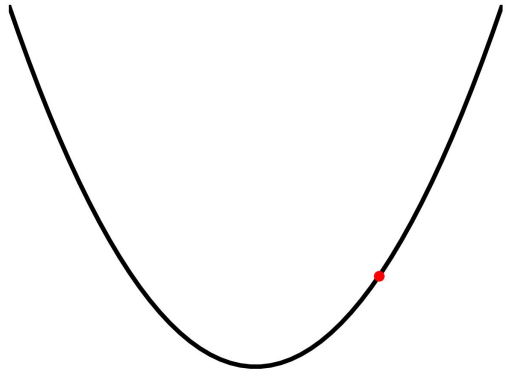
So does classical optimization theory explain the convergence of gradient descent for deep nets?

Why can't we go higher?

- gradient descent oscillates if the curvature (L) is too high!
- consider $f(x) = \frac{1}{2}Lx^2$:



$\eta < 2/L$



$\eta > 2/L$

What about in deep learning?

Can measure *local* curvature or **sharpness** by taking the top eigenvalue of the Hessian $\nabla^2 f(w)$ at parameter w:

$$L(w) = \lambda_1\left(\nabla^2 f(w)\right)$$

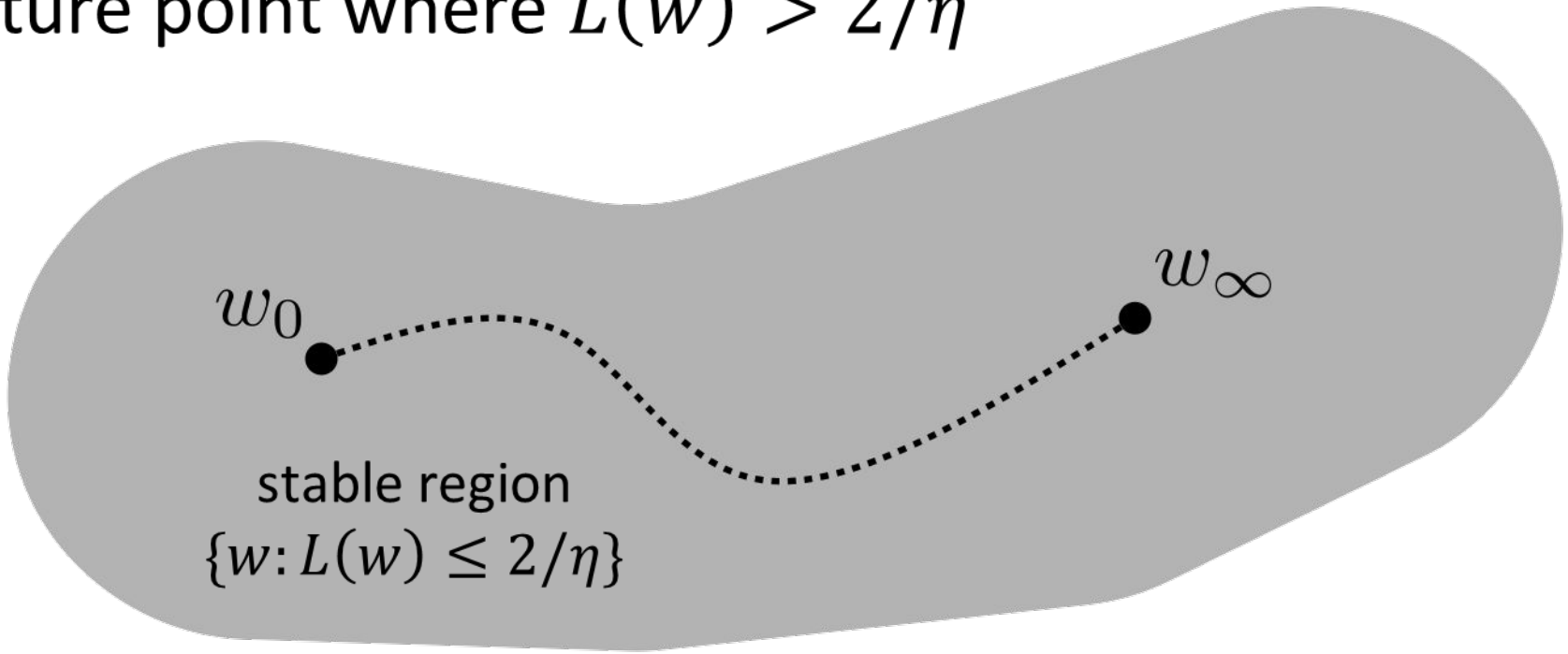
According to classical optimization theory:

- if GD is at a point x in the parameter space, it will start behaving poorly if using a step-size $\eta > 2/L(w)$
- since GD works on deep nets, this suggests it never reaches a high-curvature point where $L(w) > 2/\eta$

Expected behavior of gradient descent

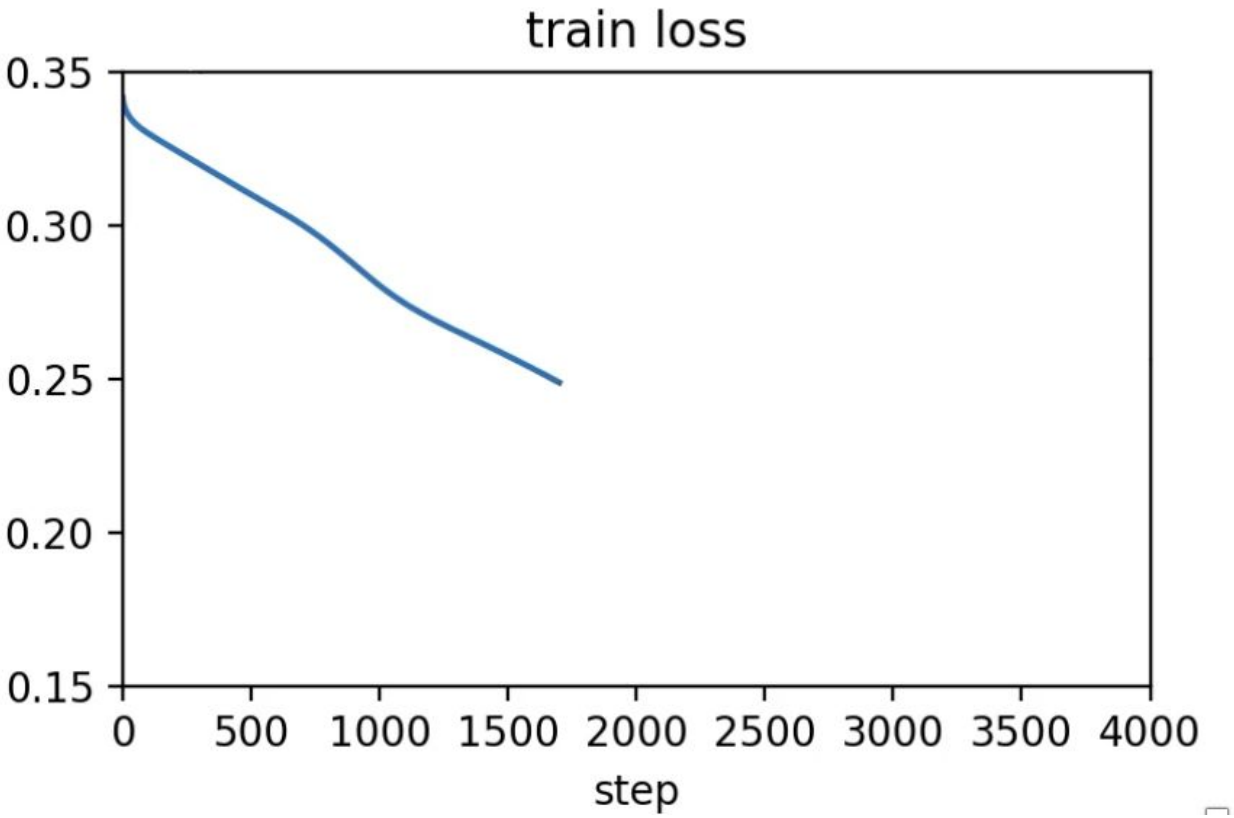
According to classical optimization theory:

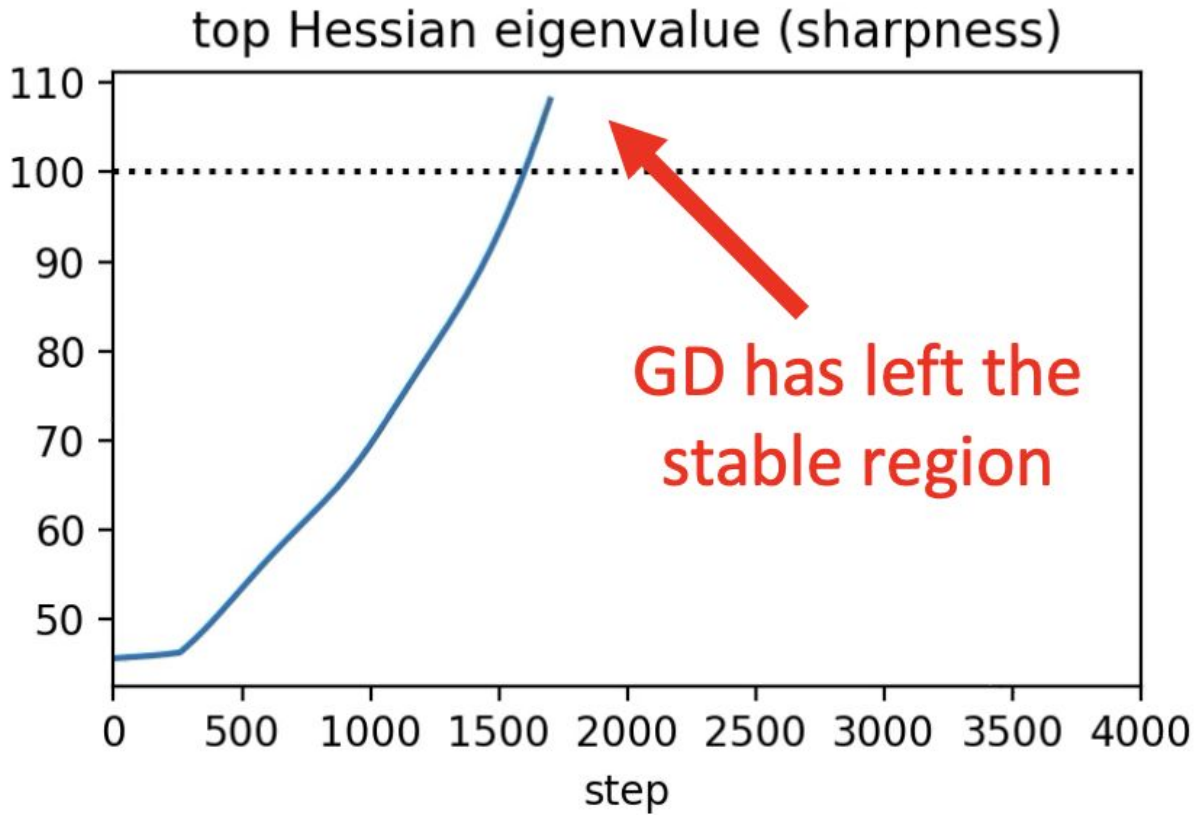
- if GD is at a point x in the parameter space, it will start behaving poorly if using a step-size $\eta > 2/L(w)$
- since GD works on deep nets, this suggests it never reaches a high-curvature point where $L(w) > 2/\eta$



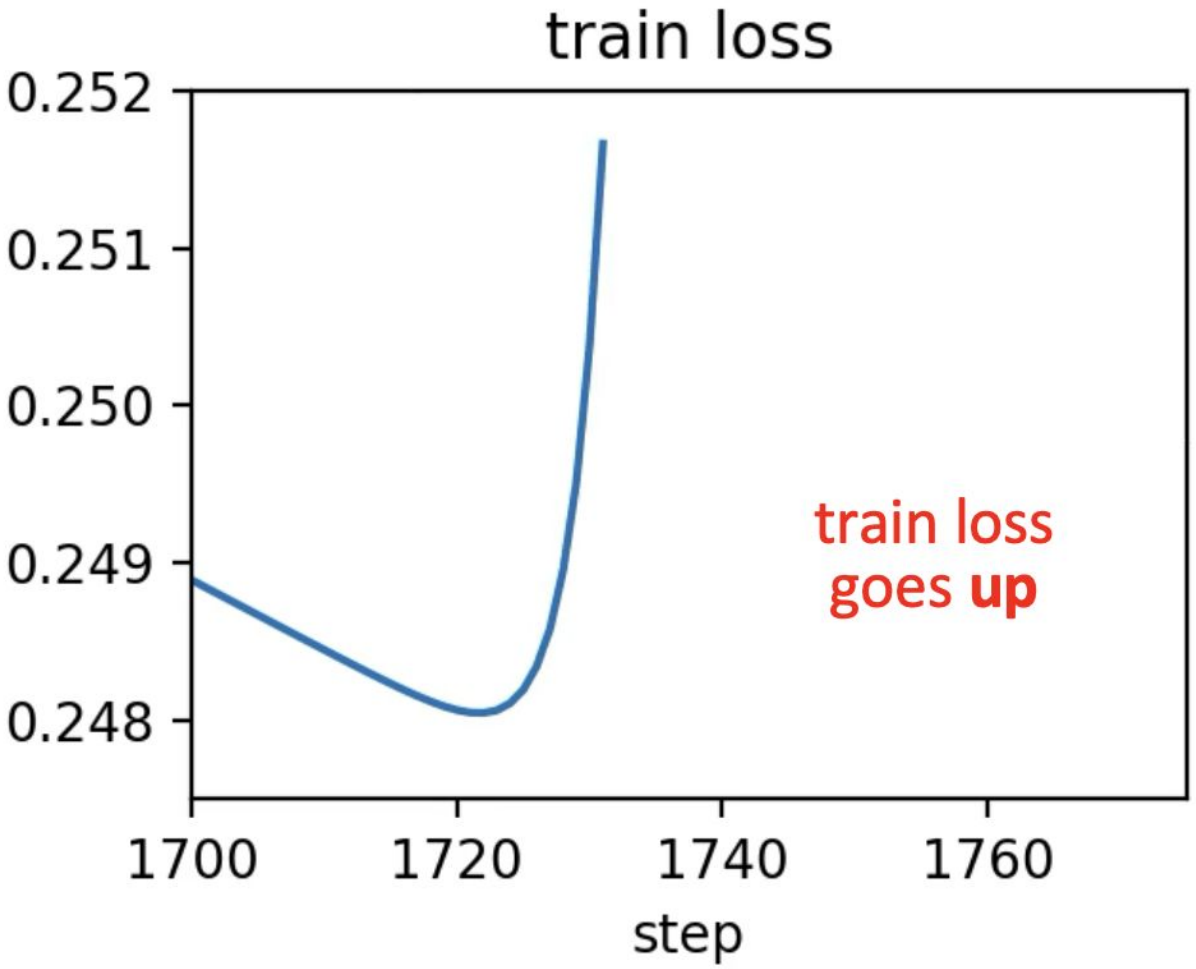
What actually happens?

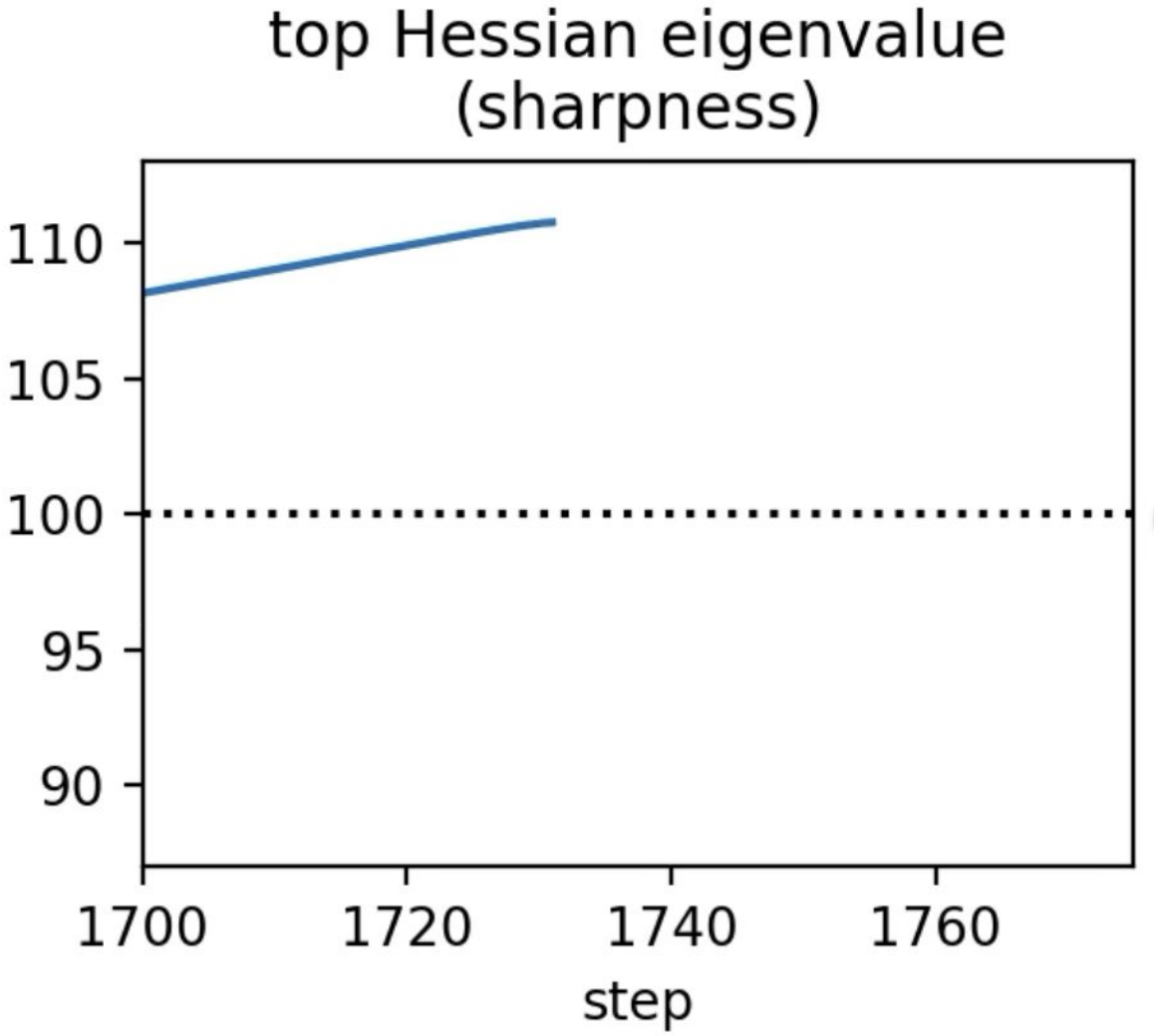
training a neural network using GD with $\eta = 0.02$
(Vision Transformer on CIFAR-10)



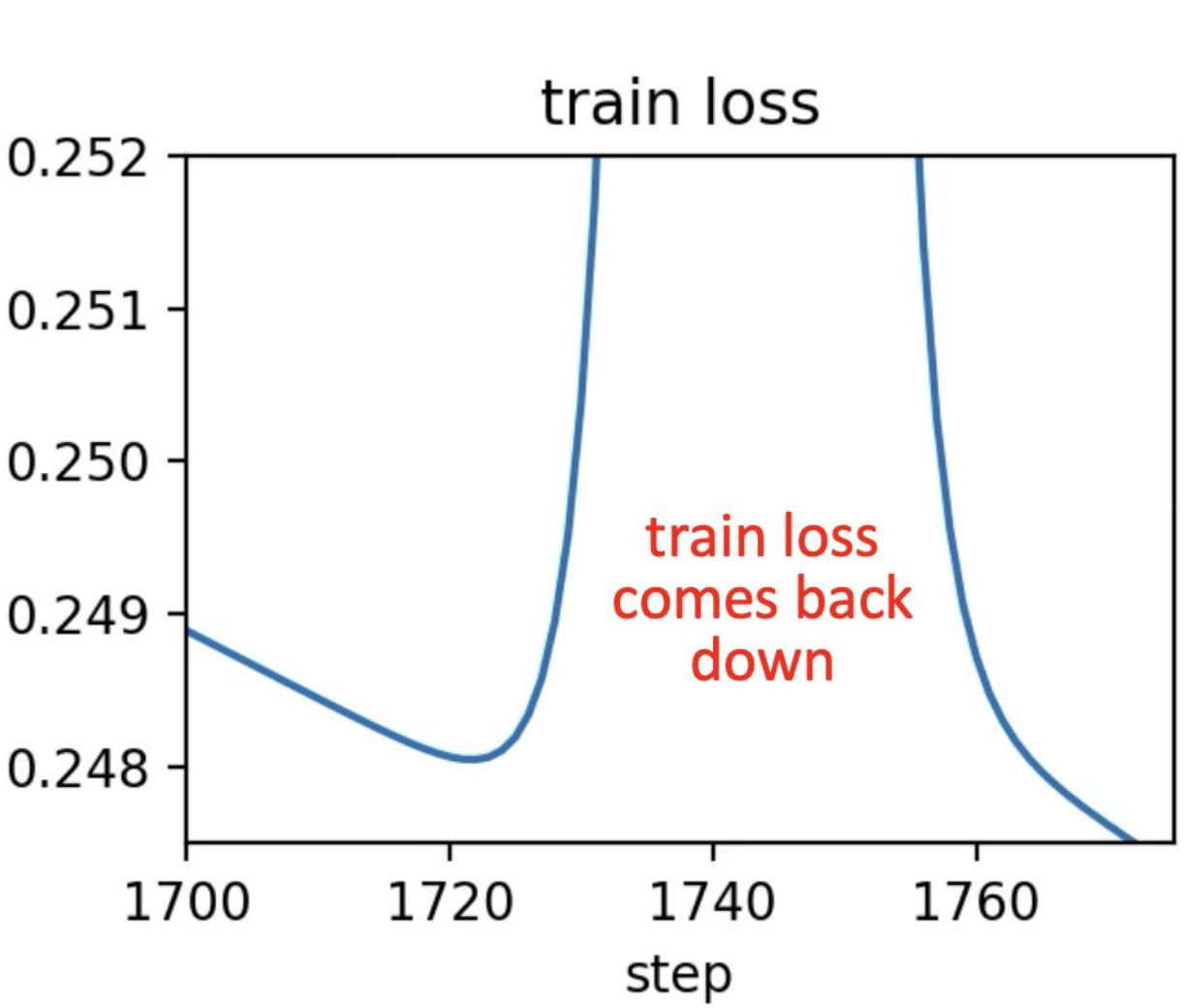


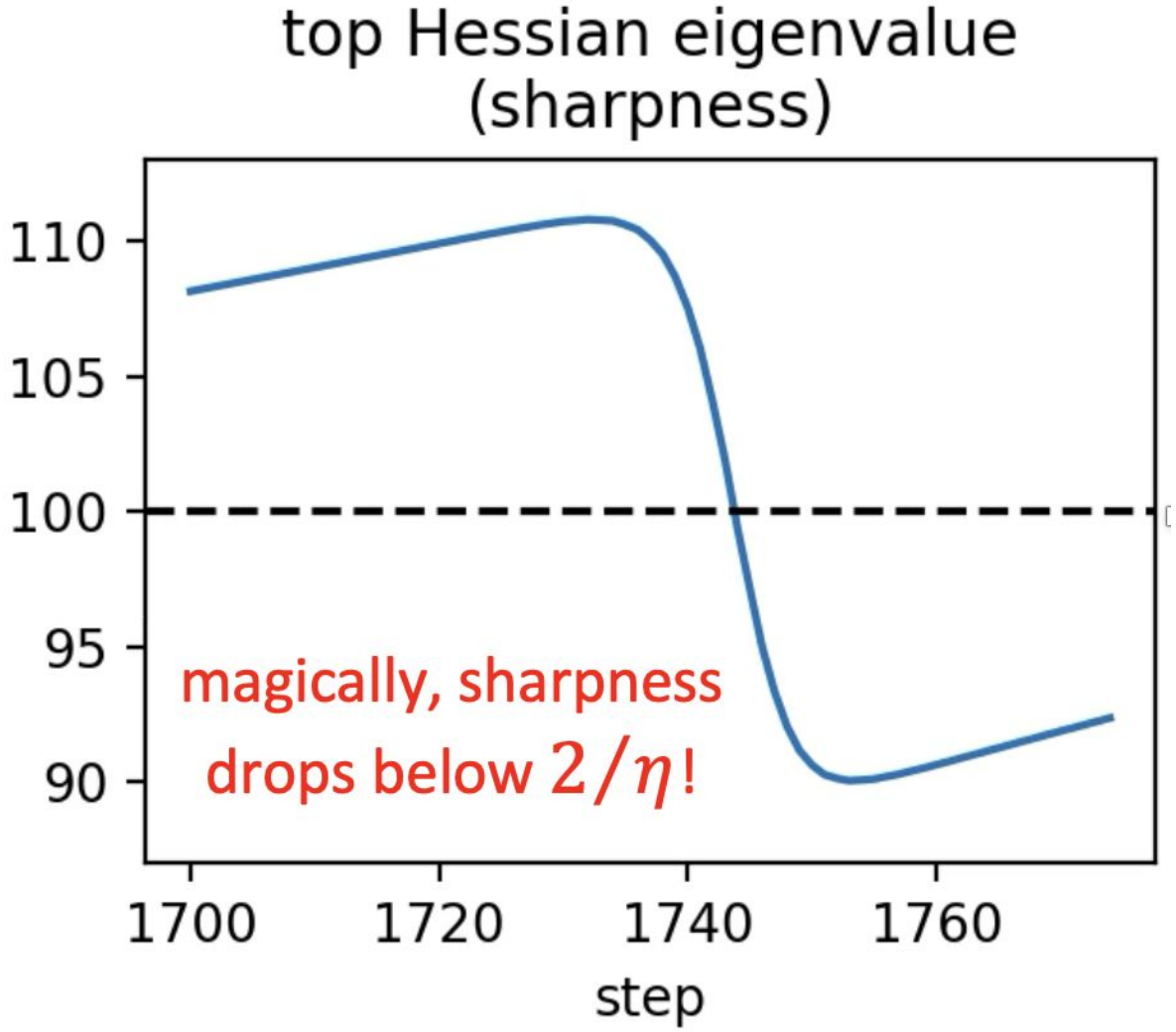
What happens next?



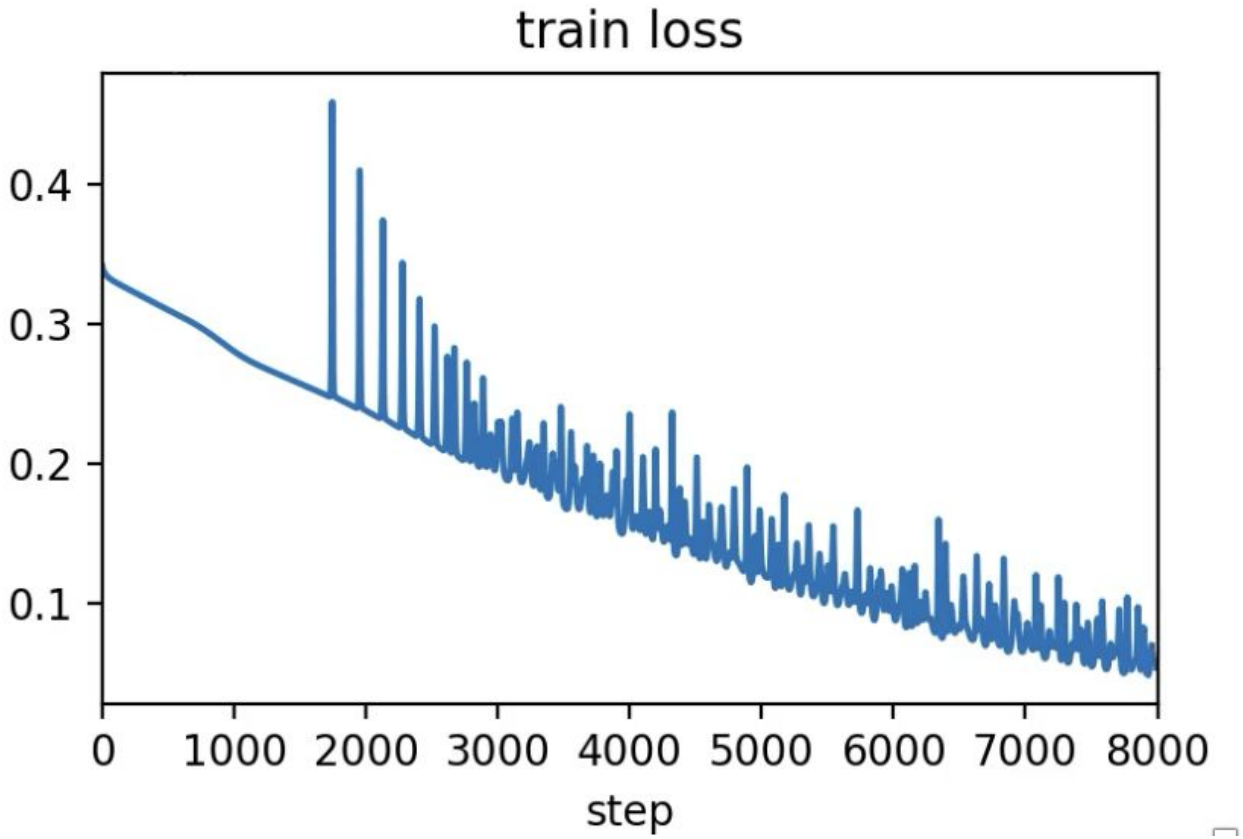


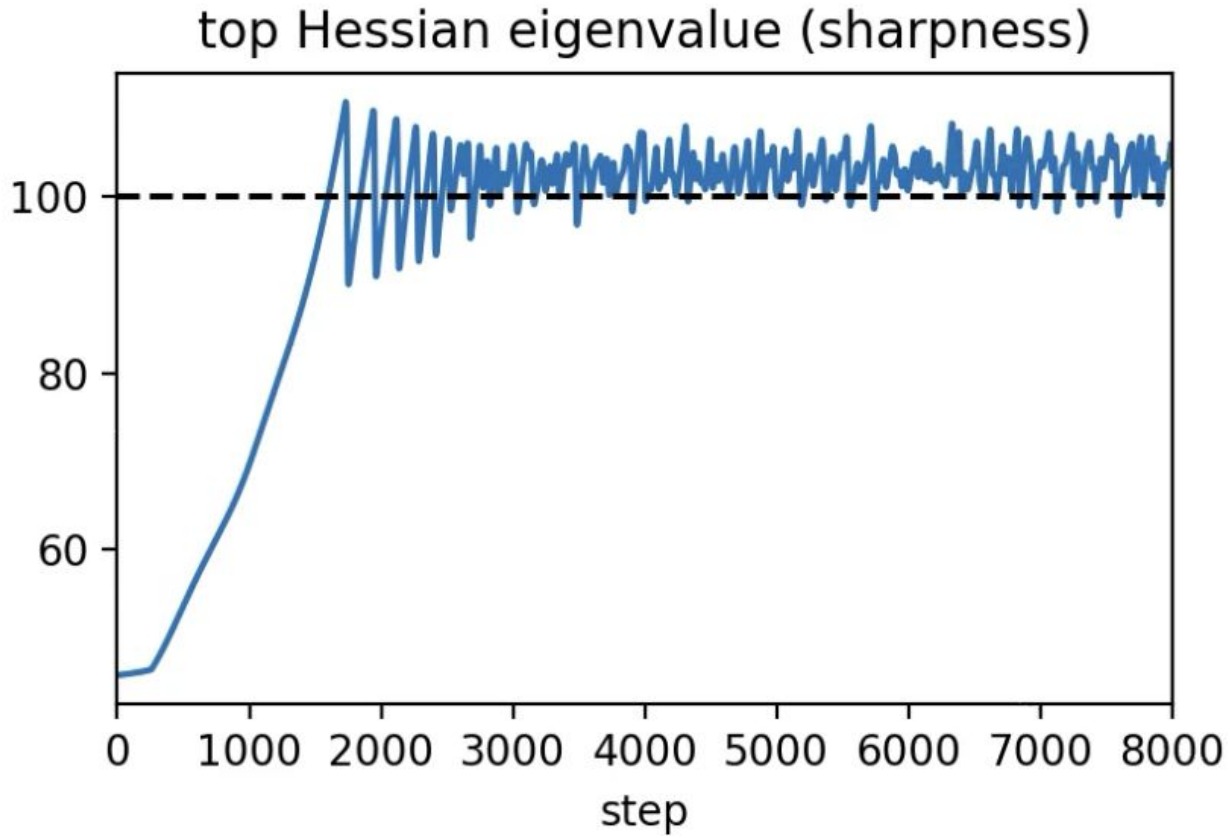
What happens next?





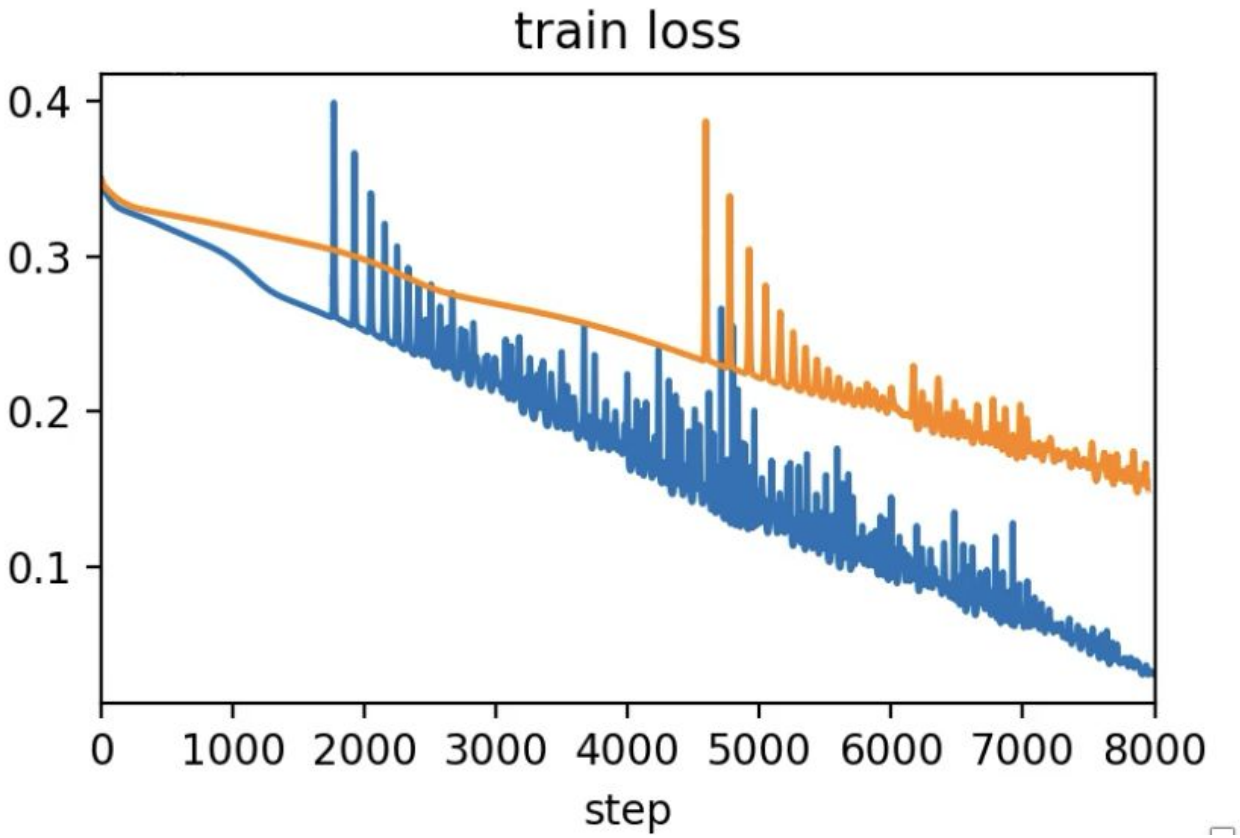
Full gradient descent trajectory

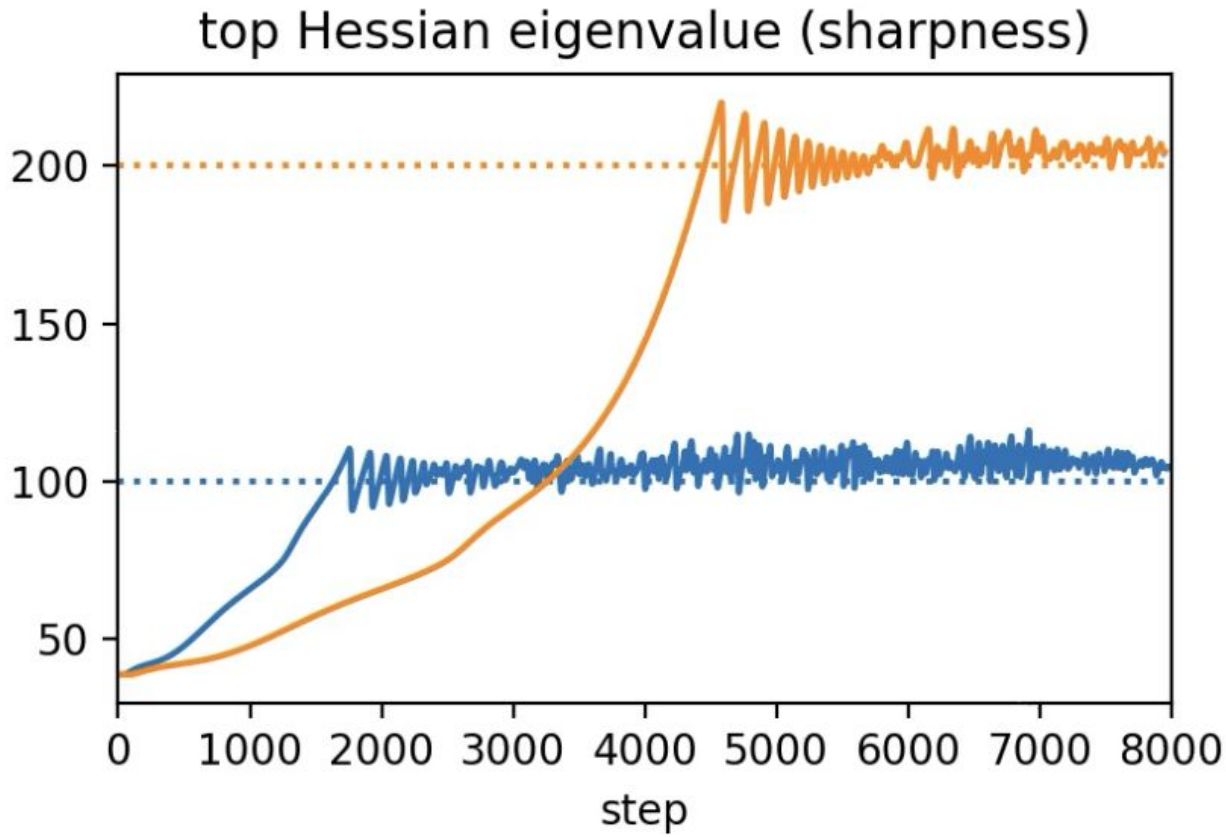




- loss goes down non-monotonically
- sharpness equilibrates around $2/\eta$

What if we train at a different learning rate?

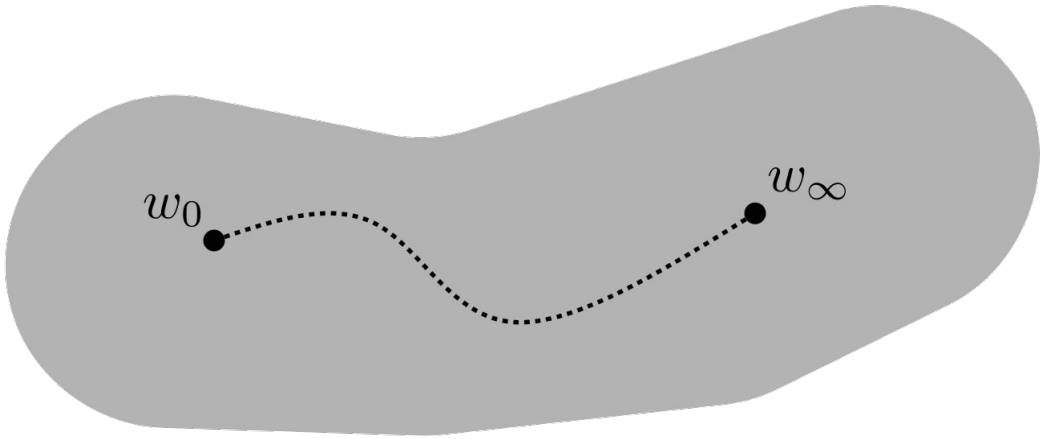




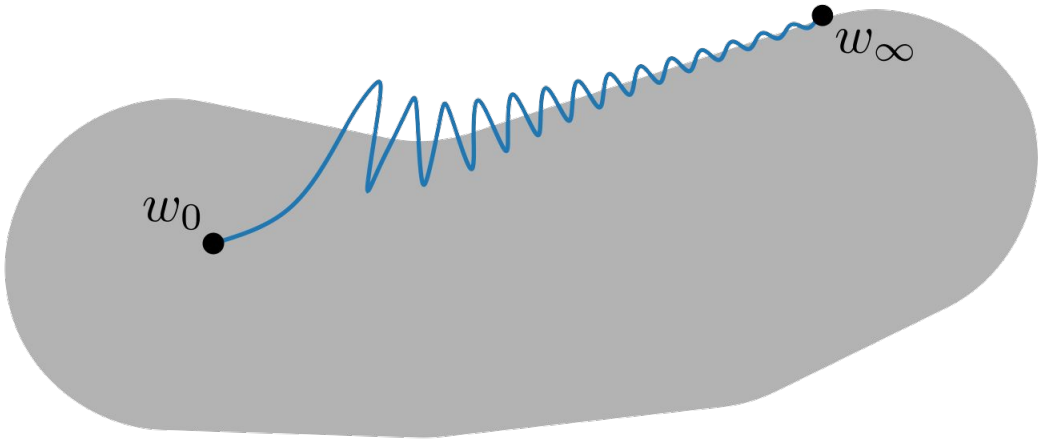
same network, smaller learning rate $\eta = 0.01$

Expectation vs. reality

expectation

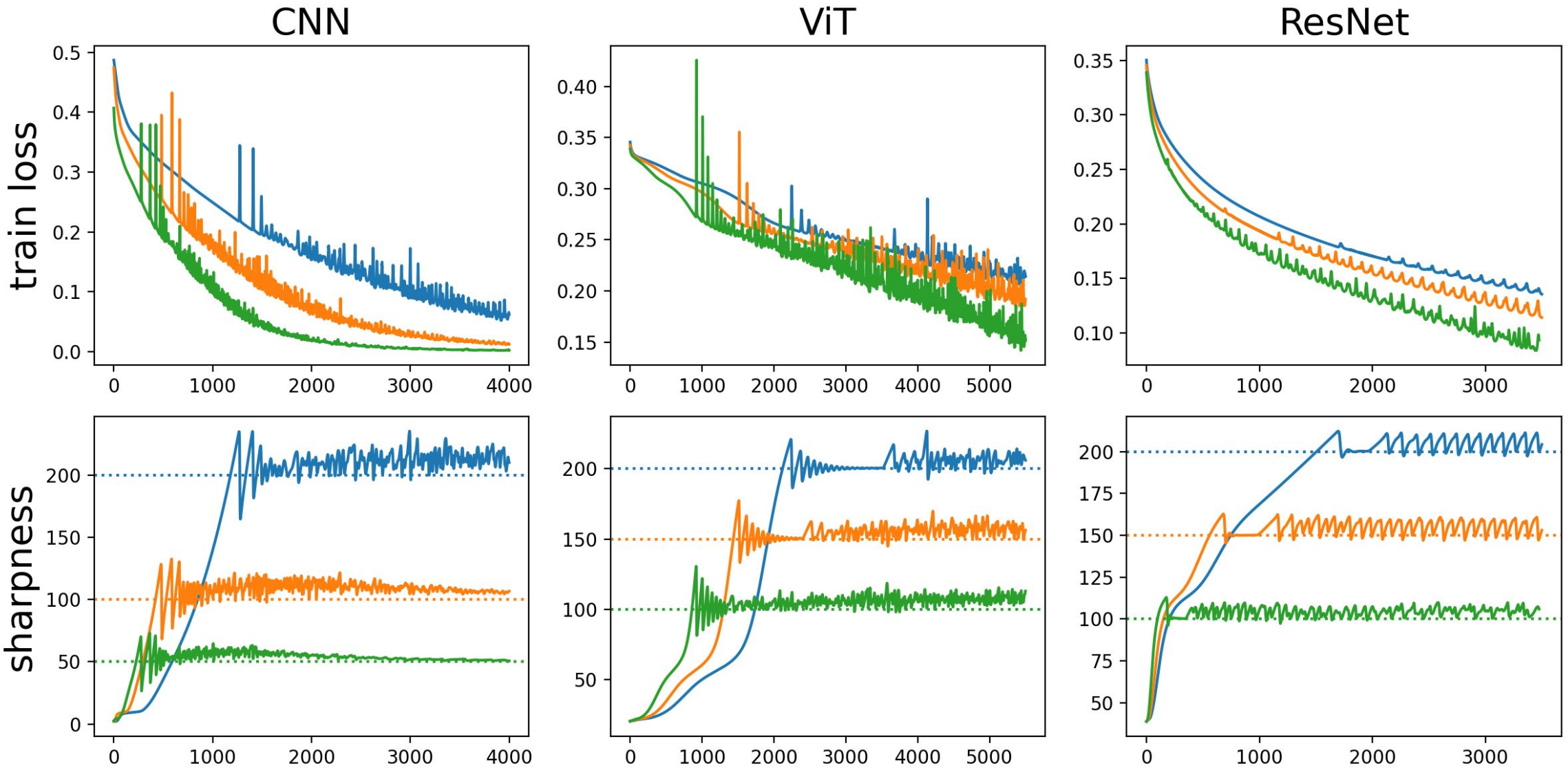


reality

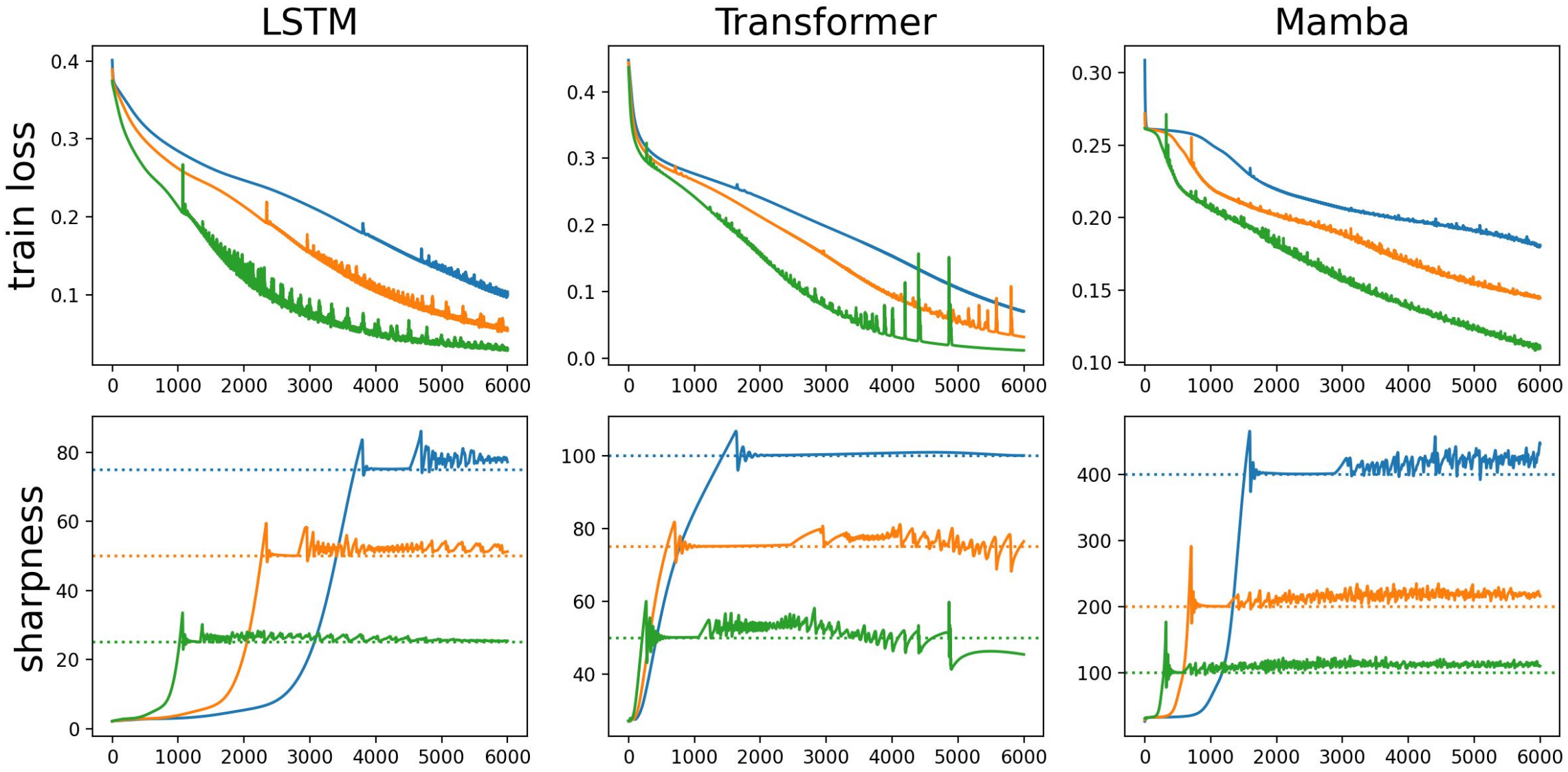


gradient descent trains at the **edge of stability**
(Cohen et al., 2021)

This behavior is generic across neural networks



This behavior is generic across neural networks



What is the takeaway?

we **always** reach a point where the smoothness is too high for the theory to be valid, i.e. where $\eta > 2/L$

- classical theory fails to explain performance of GD applied to deep nets

- we can't use it to pick learning rates!

Outline

- Limitations of traditional learning theory
  - fitting random labels, double descent
- Limitations of optimization theory
  - edge of stability
- Towards a predictive science of deep learning
  - the central flow, sparse parity, scaling laws

So what is there to do in learning theory?

Two paths:

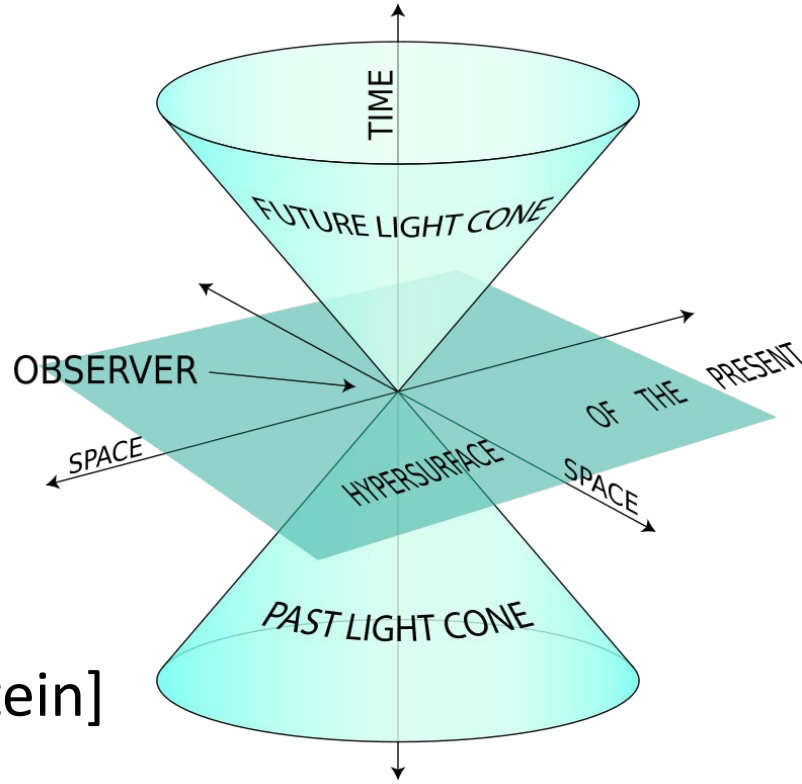
1. build out traditional theory because
   - it's mathematically elegant
   - may be informative in some contexts

2. develop a **scientific theory**

What is a **scientific** theory?

How do **physicists** develop mathematical laws about **complex systems**?

1. hypothesize
   - "light moves through a luminiferous aether" [Boyle]
2. run experiments
   - "we can't find the aether" [Michelson-Morley]
3. develop a law of nature
   - "the speed of light is the same for all observers" [Einstein]
4. derive predictions and test them
   - "light perpendicular from the source velocity is redshifted" [Ives-Stillwell]



What is a **scientific** theory?

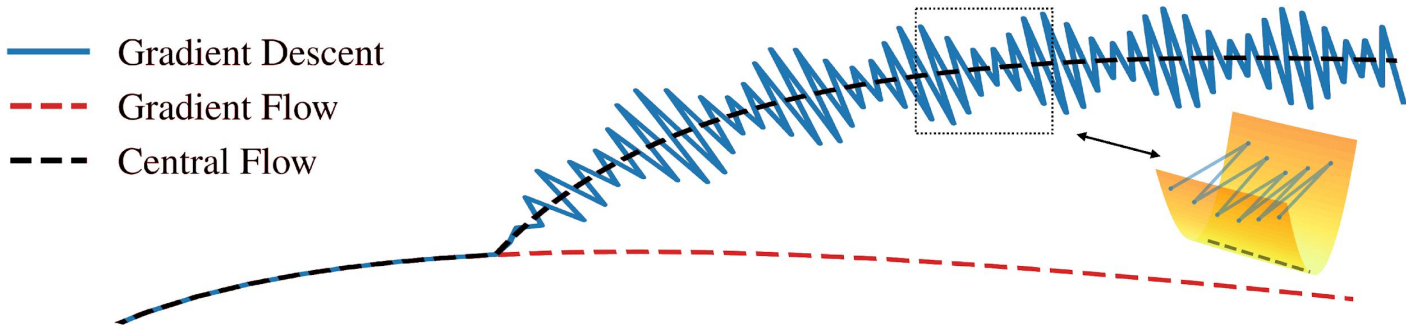
How can **machine learning researchers** develop mathematical laws about **complex systems** (neural networks)?

1. hypothesize
   - "gradient descent stays in a stable region of the parameter space"
2. run experiments
   - "the gradient descent trajectory leaves the stable region"
3. develop a law of nature
   - "the gradient descent trajectory on a CNN hovers around sharpness = $2/\eta$"
4. derive predictions and test them
   - "the gradient descent trajectory on Mamba does the same"

The modern science of deep learning

many observed phenomena:

- double descent
- edge of stability
- grokking (OpenAI)
- mode connectivity (Garipov et al.)
- …



fewer mathematical theories:

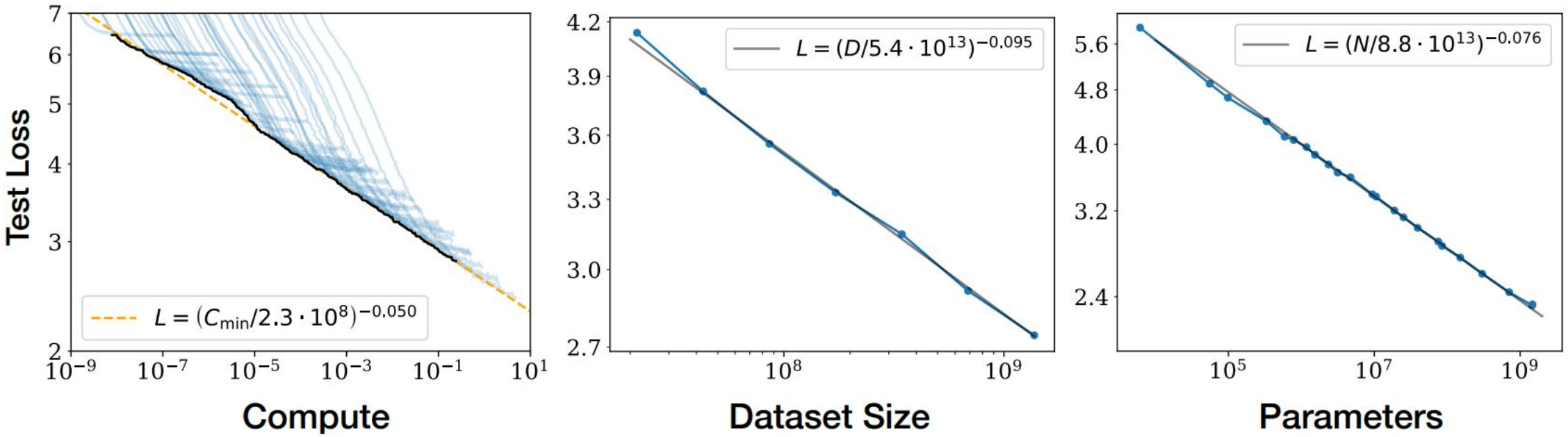
- the central flow (Cohen et al.): a differential equation that predicts the time-averaged trajectory of gradient descent
- scaling laws (Kaplan et al.): power laws predicting the accuracy of an LLM as a function of model size, dataset size, and compute

What is a scaling law?

Suppose we want to train a massive Transformer on Internet-scale data.

Q: how does the loss depend on compute (C), dataset size (D), and number of model parameters (N)?

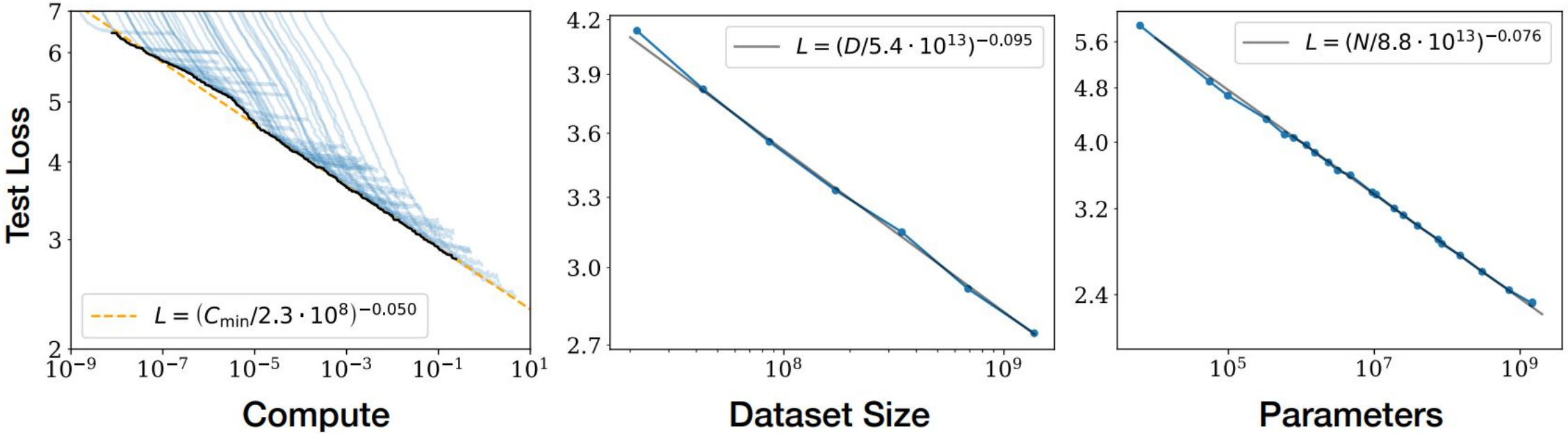
A: as power laws in the individual quantities



OpenAI

Implications of scaling laws

1. need to simultaneously increase the model and the dataset size
2. predictive scaling law let us derive compute-optimal allocations before we train the model
3. many remaining questions around the right batch size to use, multi-scale hyperparameter tuning, etc.



OpenAI

Summary

1. mathematical analysis of learning is still relevant
   - even in industrial settings
   - if it has empirical backing

2. however, doing such science for deep learning can require
   - running thousands of experiments
   - rigorous results tracking
   - industrial scale resources (e.g. scaling laws)
   - an awareness that results may be made irrelevant by the rapid advances in the field

Thanks Everyone!

Slides in this lecture were created by Misha Khodak with some materials adapted from Jeremy Cohen.