Carnegie Mellon University

FLATIRON INSTITUTE

How Does Gradient Descent Work?

Part I of a two-part talk with Alex Damian

Jeremy Cohen · ELLIIT, Lund · May 8, 2026

Introduction

- Rule of thumb: understand simple things before complex things
- What's the simplest optimization algorithm we don't yet understand (in DL)?
- Claim: it's gradient descent
  - First, I'll describe what's wrong with the traditional understanding
  - Then, I'll explain how to fix it
  - Finally, I'll cover the broader point (i.e. beyond GD) to take away

Gradient descent

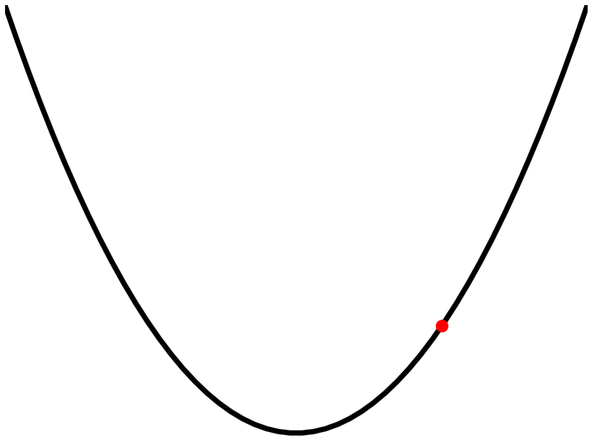
- Consider gradient descent with a fixed learning rate η:

$$w_{t+1} = w_t - \eta \nabla L(w_t)$$

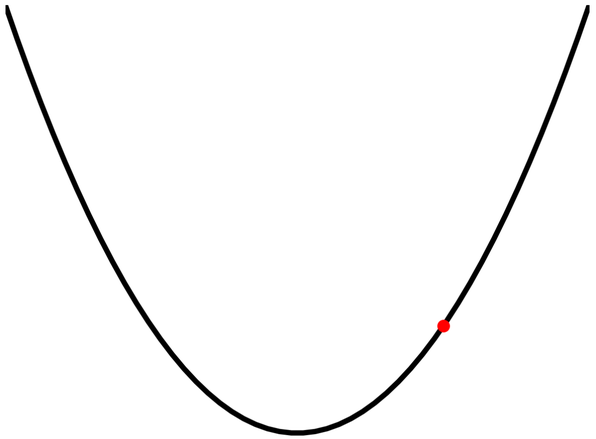
- This is the simplest first-order optimizer, so we must understand it first

Background: quadratic objective functions

- On quadratics, GD oscillates if the *curvature* (2nd derivative) is too high
- Consider a 1d quadratic function $L(x) = \frac{1}{2}Sx^2$, with curvature $L''(x) = S$



$S < 2/\eta$



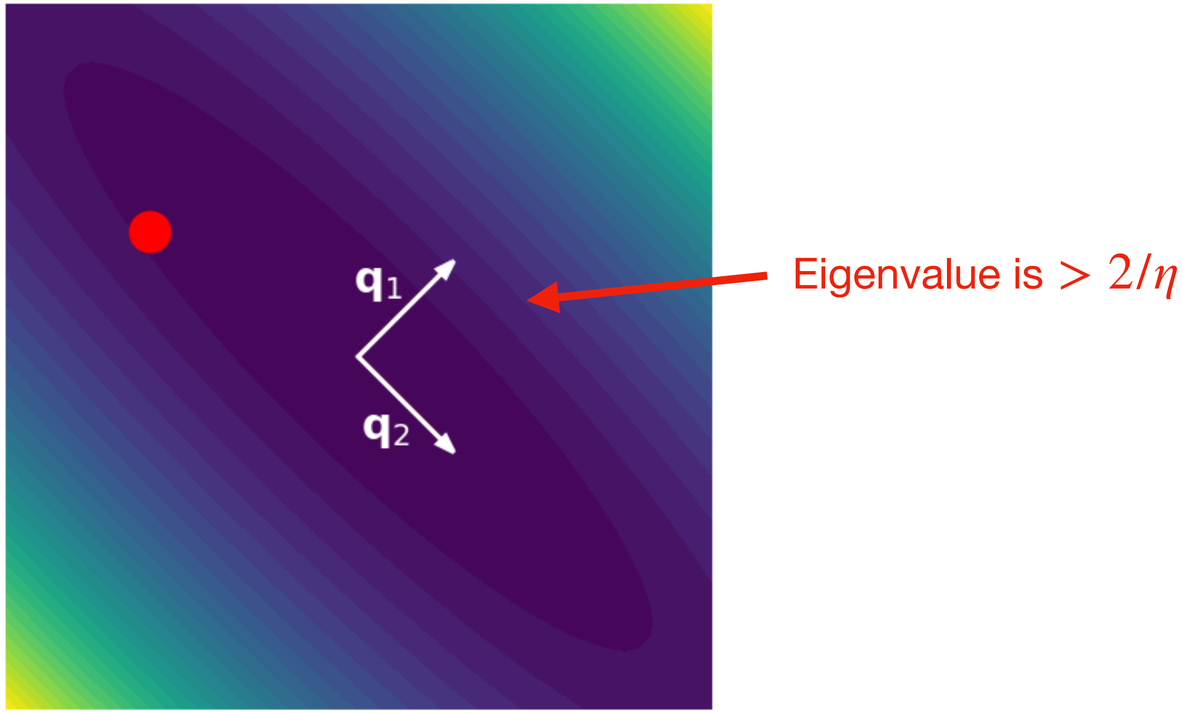
$S > 2/\eta$

Background: quadratic objective functions

- For a quadratic in *multiple* dimensions, curvature is quantified by Hessian
- GD oscillates along Hessian eigenvectors with eigenvalues greater than $2/\eta$

Background: quadratic objective functions

- For a quadratic in *multiple* dimensions, curvature is quantified by Hessian
- GD oscillates along Hessian eigenvectors with eigenvalues greater than $2/\eta$



What about deep learning?

- For DL objectives, can take quadratic Taylor approximation around any w
- Dynamics of GD on this quadratic depend on the top eigenvalue of the Hessian $H(w)$, i.e. the *sharpness* $S(w) := \lambda_1(H(w))$
- If sharpness $S(w) > 2/\eta$, GD would diverge on the quadratic Taylor approximation
- This suggests that GD doesn't function properly if sharpness $S(w) > 2/\eta$

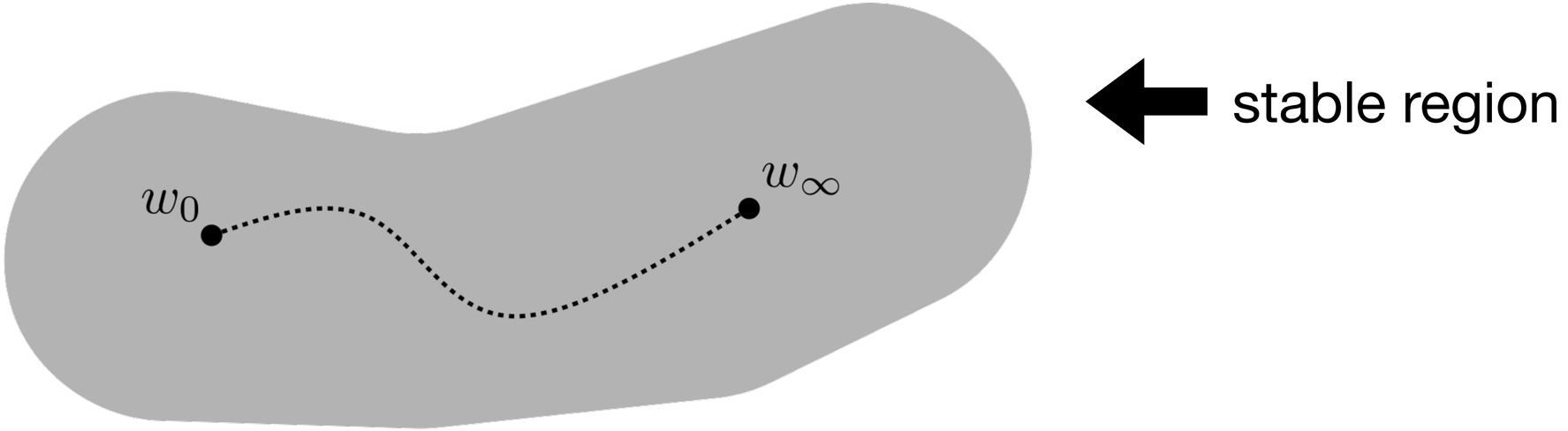
Gradient descent in deep learning

- Why does gradient descent converge in deep learning?
- Natural idea: sharpness $S(w)$ remains below $2/\eta$ throughout training
  - i.e. GD stays inside the "stable region" $\{w : S(w) \leq 2/\eta\}$

stable region

Gradient descent in deep learning

- Why does gradient descent converge in deep learning?
- Natural idea: sharpness $S(w)$ remains below $2/\eta$ throughout training
  - i.e. GD stays inside the "stable region" $\{w : S(w) \leq 2/\eta\}$



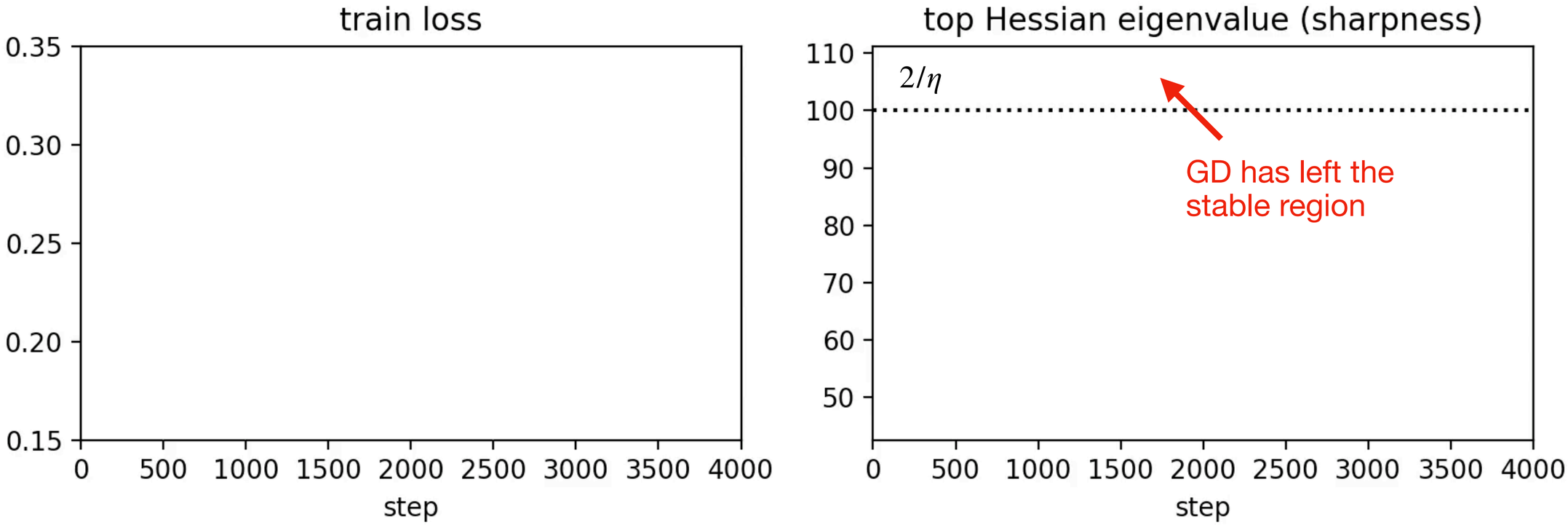
- This is the picture suggested by traditional optimization theory ("L-smoothness")

Deep learning reality

- Train neural network using GD with $\eta = 0.02$ (ViT on CIFAR-10):

Deep learning reality

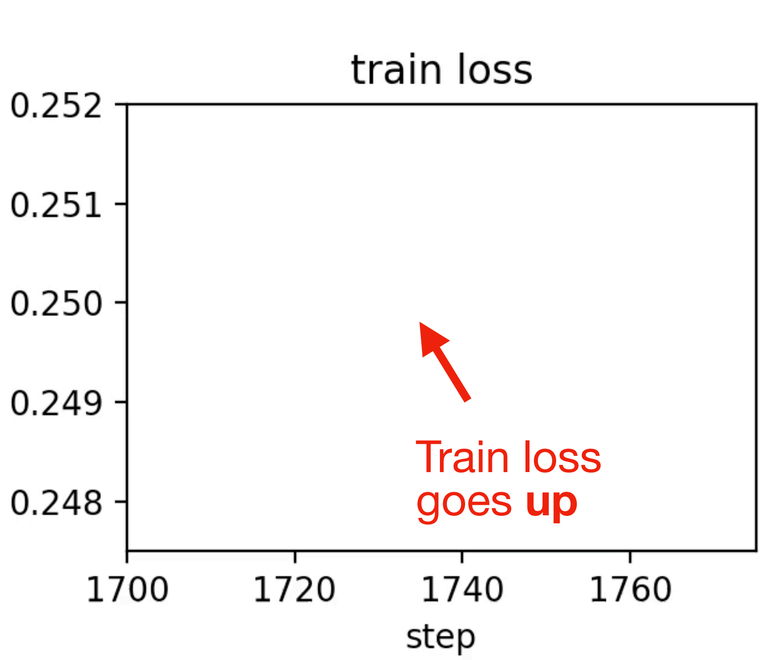
- Train neural network using GD with $\eta = 0.02$ (ViT on CIFAR-10):

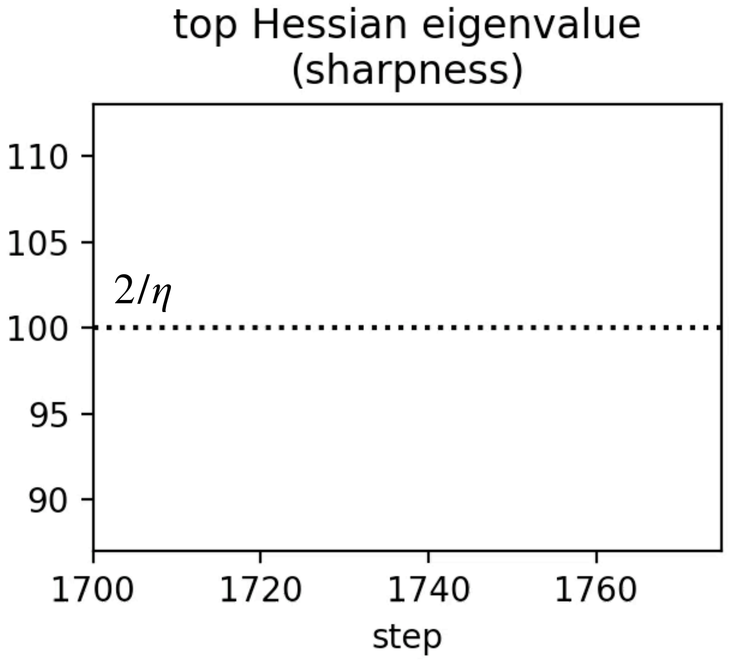


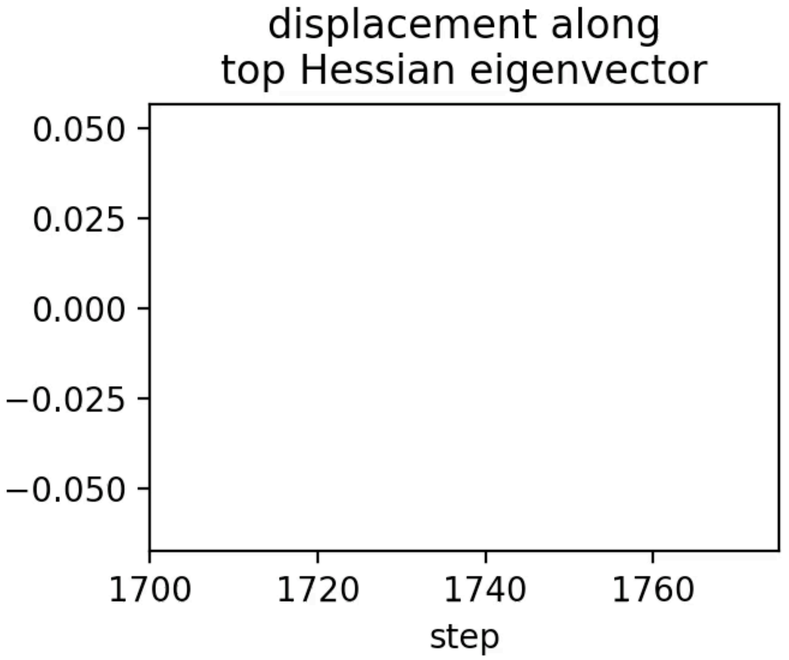
Quadratic Taylor approximation predicts growing oscillations along top Hessian eigenvector

What happens next?

$u_{t_0}^\top (w_t - w_{t_0})$ where u_{t_0} is the top Hessian eigenvector at step t_0 = 1700

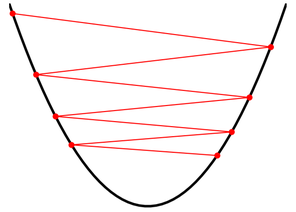






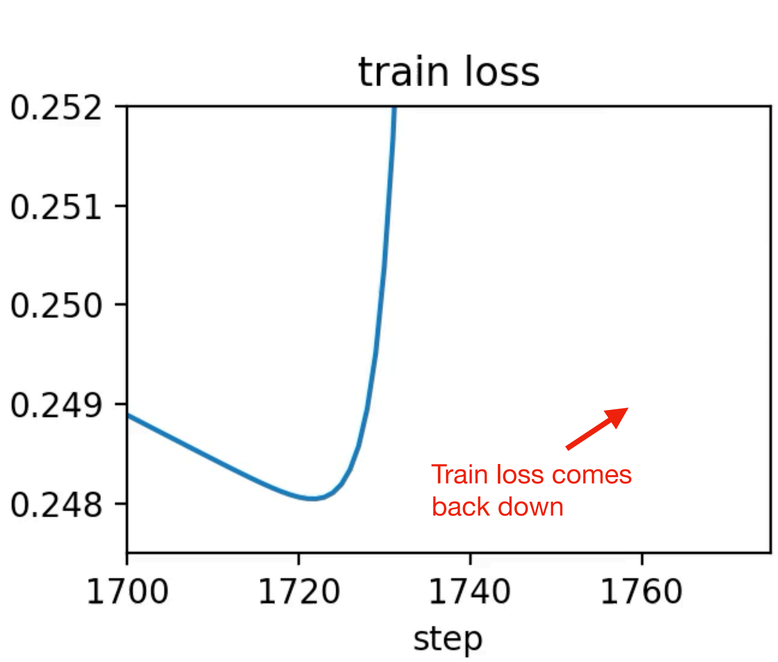
What happens next?

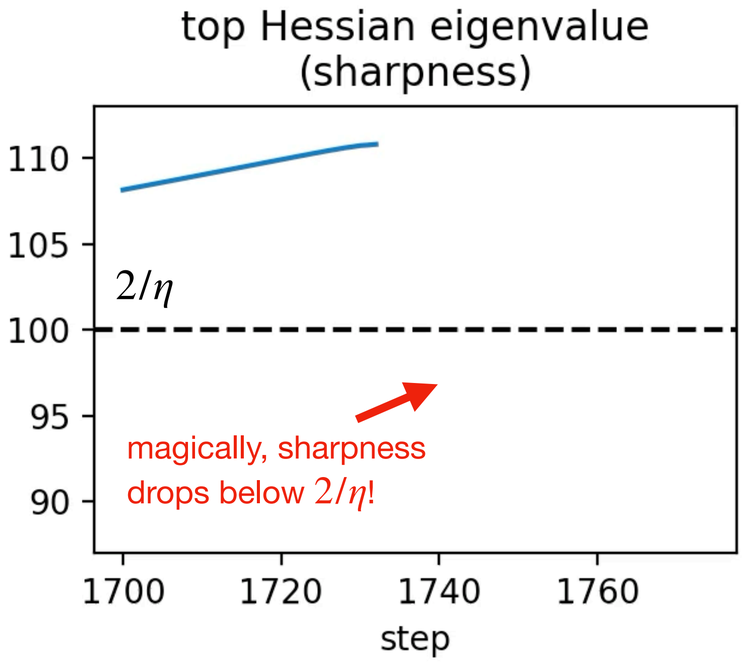
This quantity is predicted to oscillate

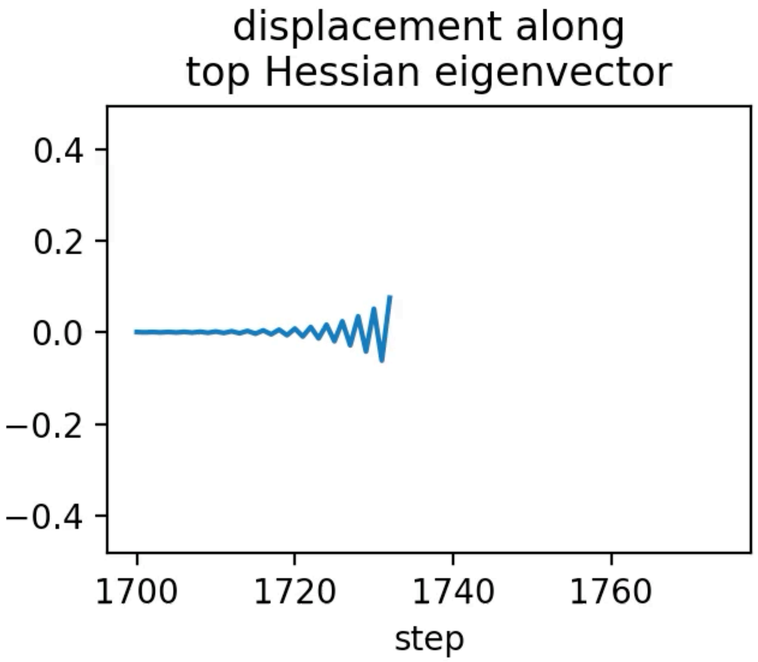


ViT / CIFAR-10 / gradient descent

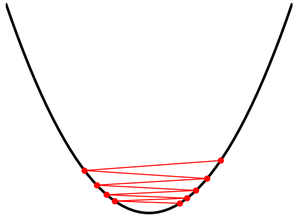
What happens next?







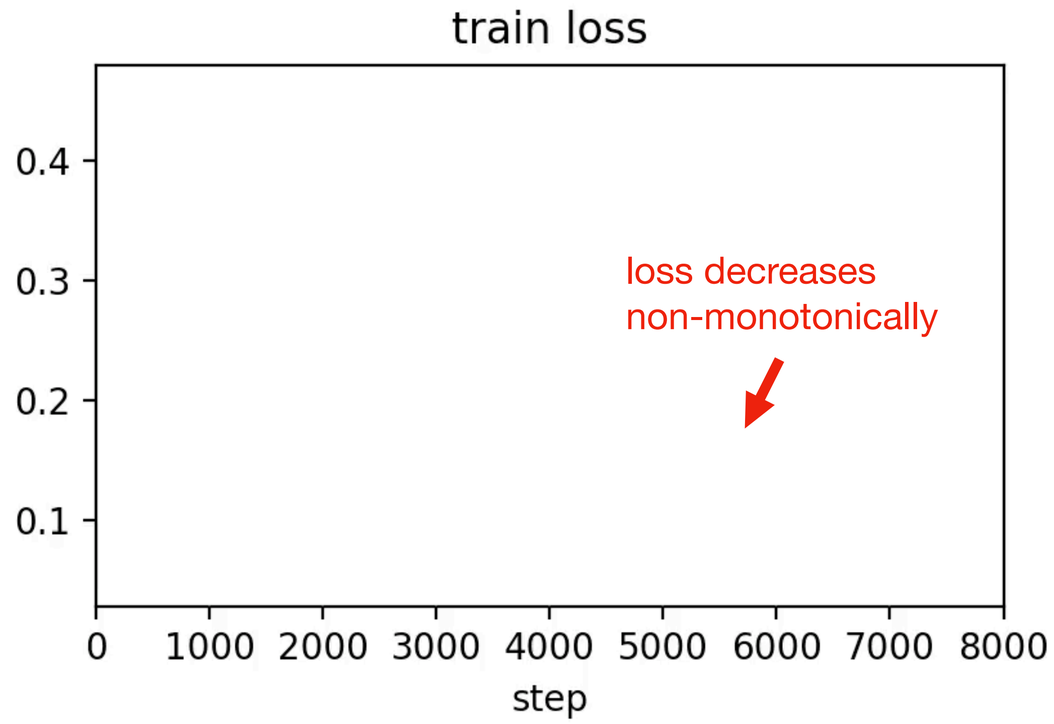
Mystery: why does the sharpness drop?

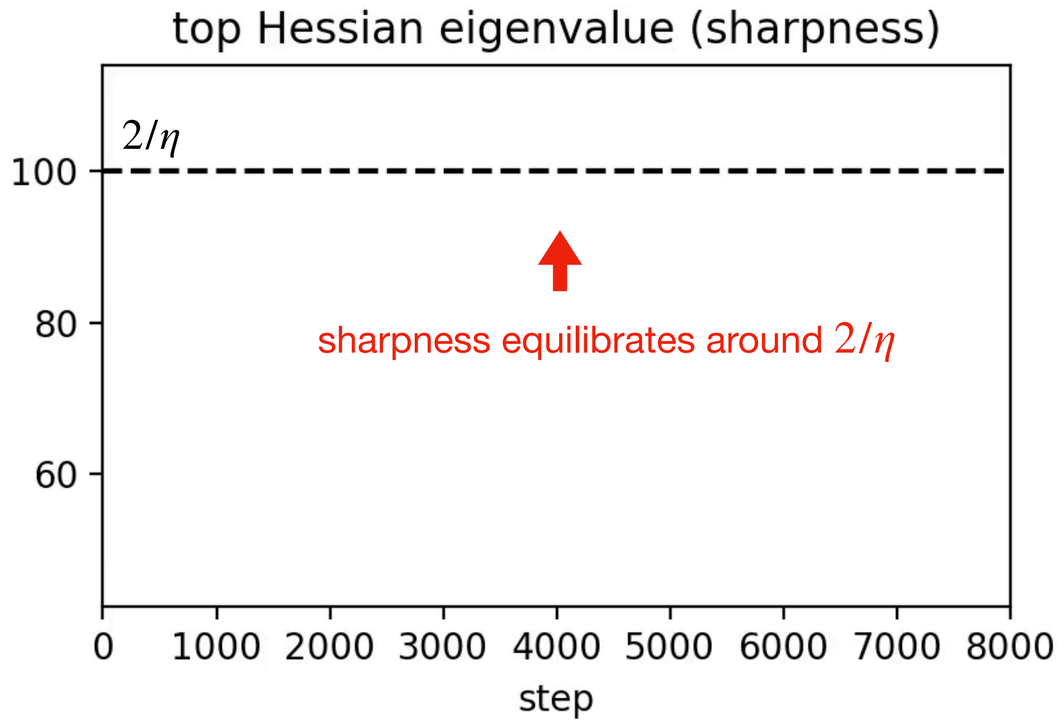


oscillations
shrink

ViT / CIFAR-10 / gradient descent

Full gradient descent trajectory

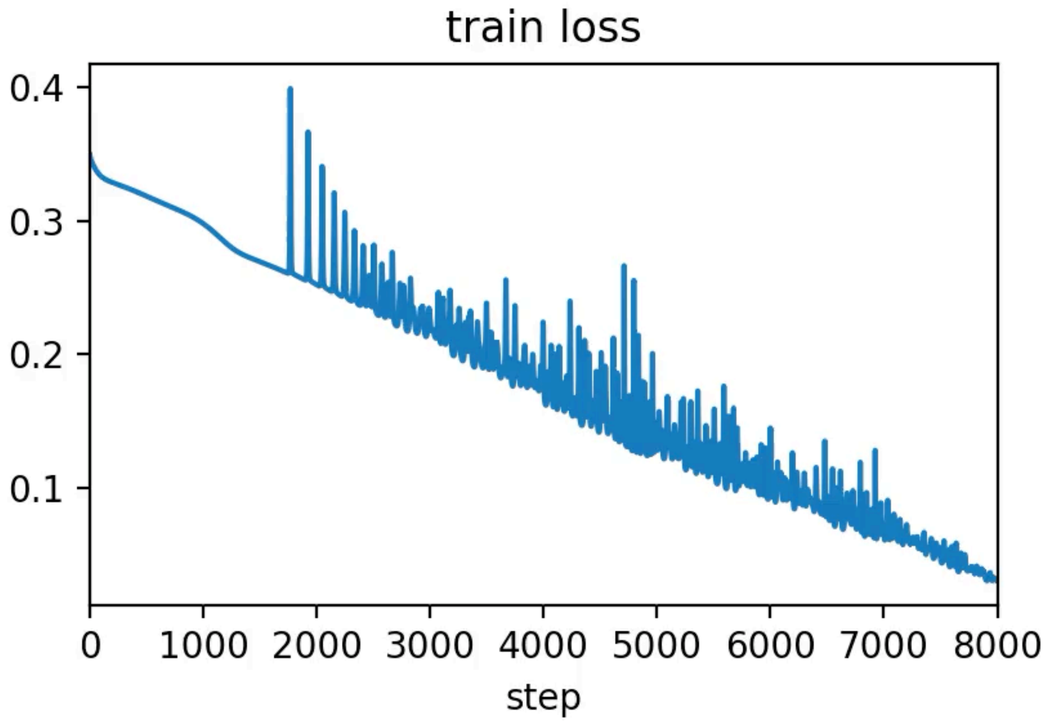


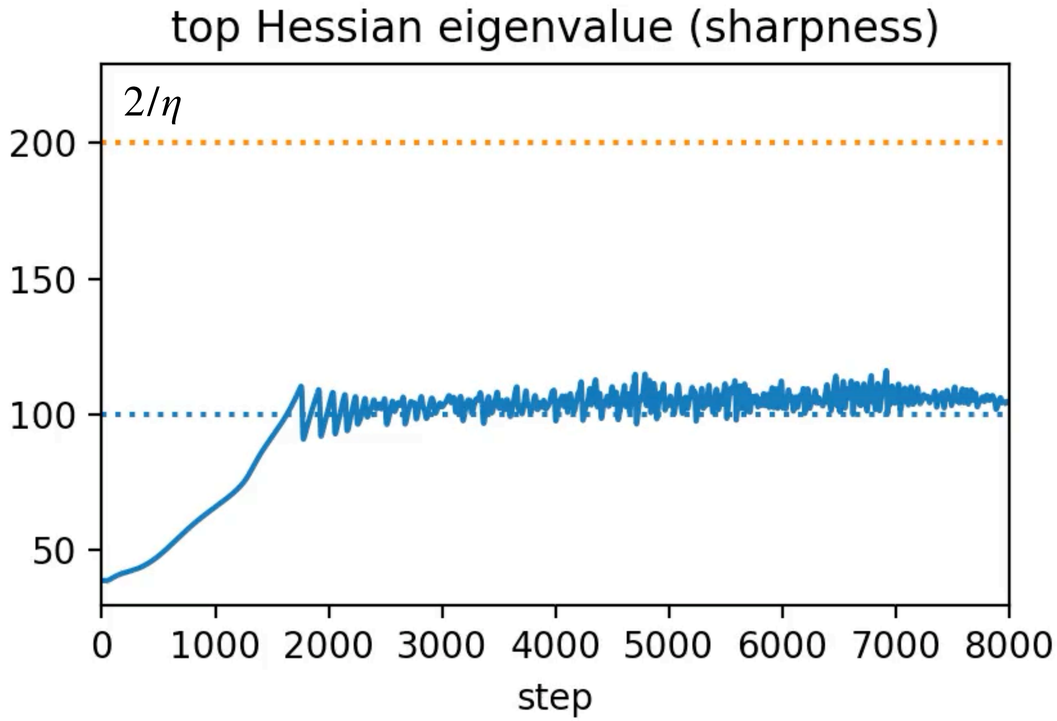


ViT / CIFAR-10 / gradient descent

What if we train at a different learning rate?

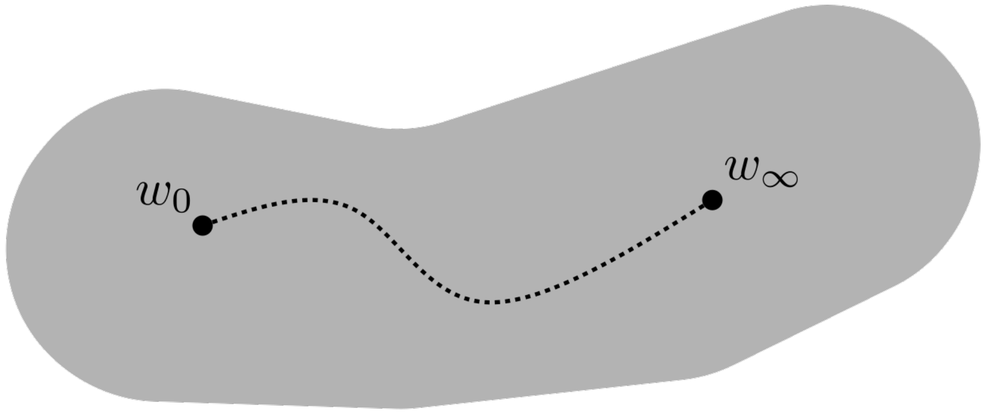
- Train same network with smaller learning rate $\eta = 0.01$ (orange):



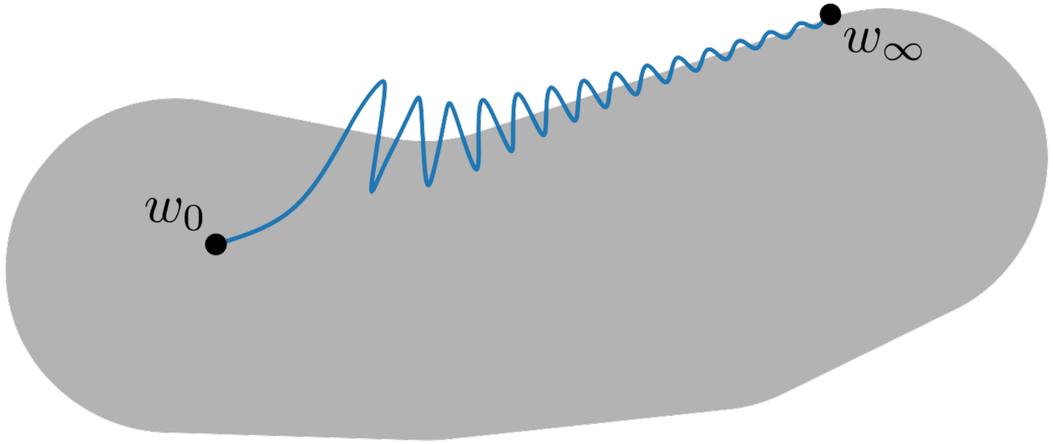


ViT / CIFAR-10 / gradient descent

Expectation vs. reality



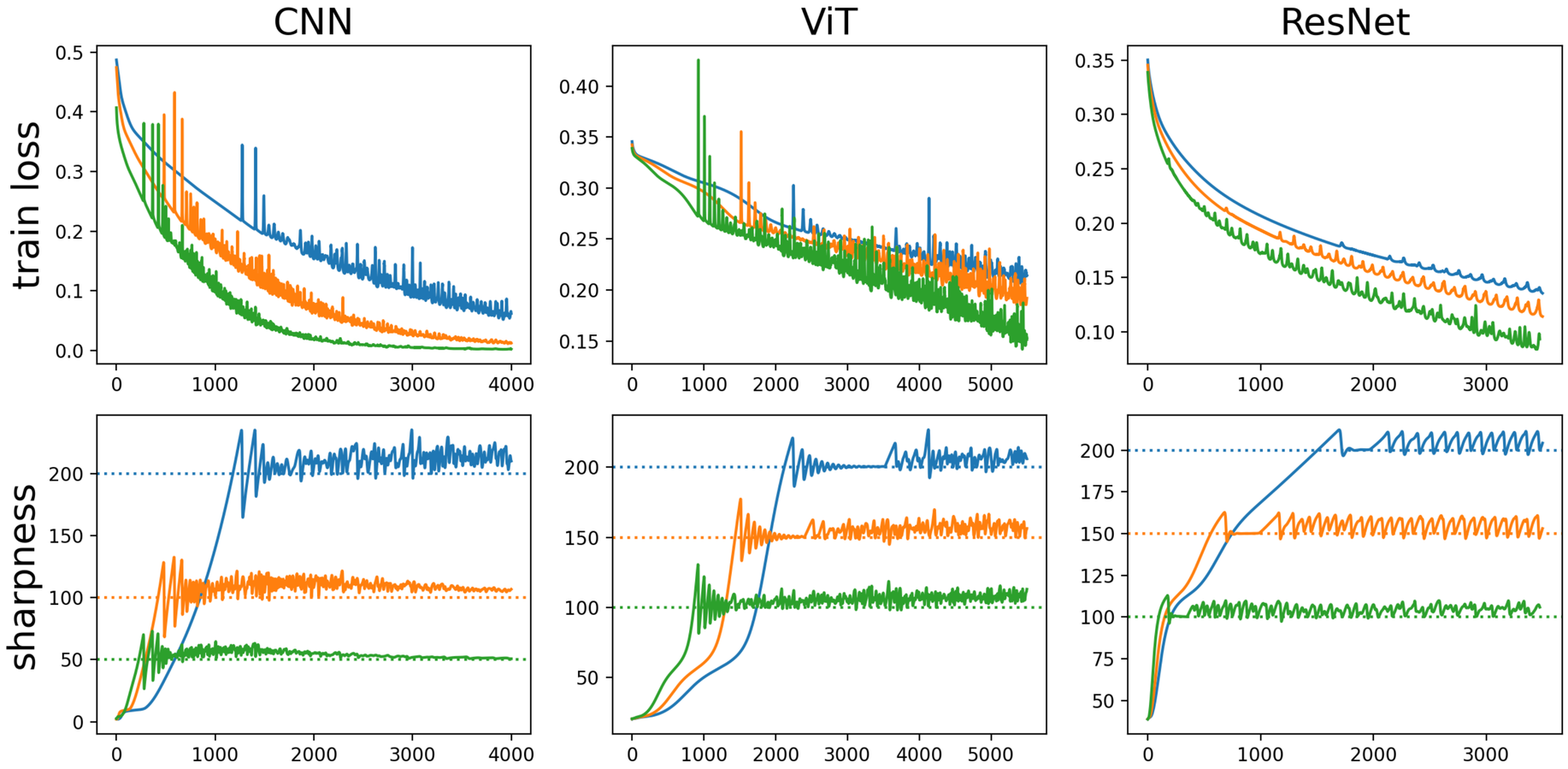
Expectation



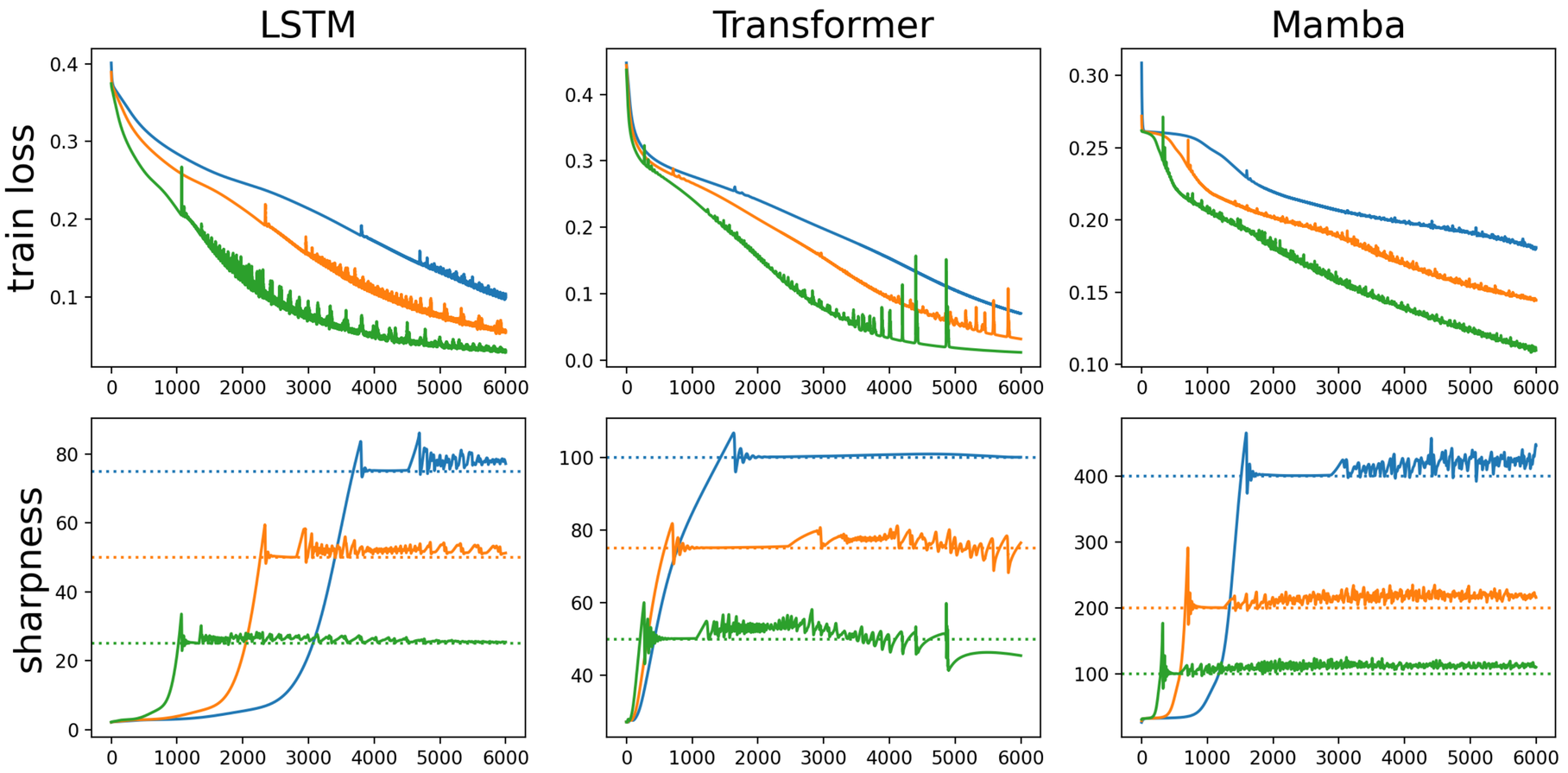
Reality

Gradient descent trains at the **edge of stability**

This behavior is generic across DL settings



This behavior is generic across DL settings



- This is not a weird edge case, it's the **typical** behavior of GD in DL

What's going on?

Cohen, Kaur, Li, Kolter, Talwalkar. *Gradient descent on neural networks typically occurs at the edge of stability*. ICLR '21.

Why does gradient descent work in deep learning?

The answer

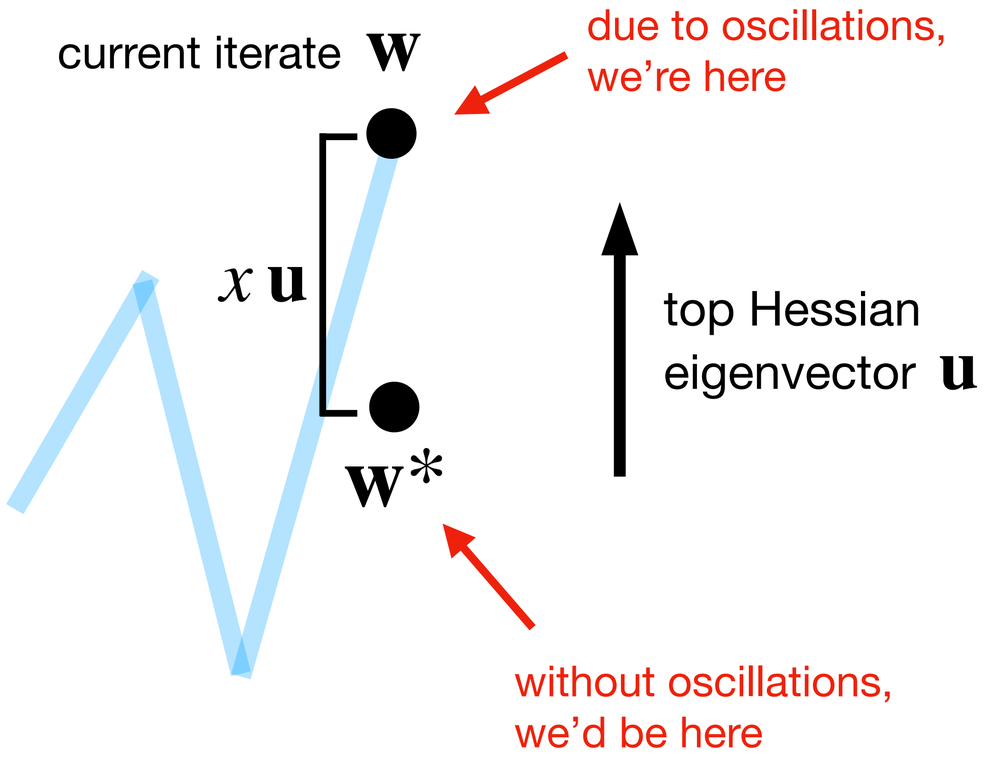


Damian*, Nichani*, Lee. *Self-stabilization: the implicit bias of gradient descent at the edge of stability.* ICLR ’23.

- To understand dynamics of GD, need to Taylor expand to *third-order*.
- This expansion reveals the key ingredient missing from traditional theory:

Oscillations along the top Hessian eigenvector automatically reduce the top Hessian eigenvalue.

Informal sketch



cartoon of weight-space dynamics

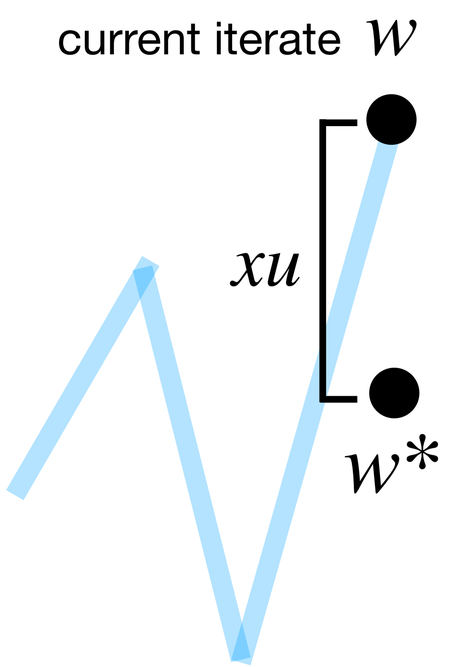
Suppose that GD is oscillating along the top Hessian eigenvector $\mathbf{u}$

How does the gradient ∇L at

$$\mathbf{w} = \mathbf{w}^* + x\,\mathbf{u}$$

relate to the gradient at w^*?

Informal sketch

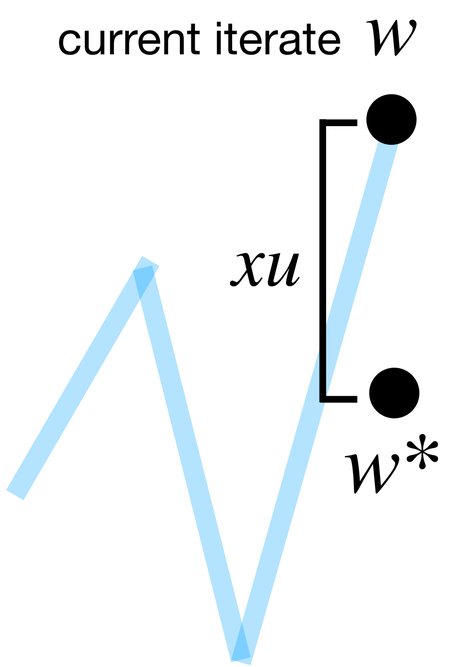


By Taylor expansion around $\mathbf{w}^*$:

$$\nabla L(\mathbf{w}^* + x\mathbf{u}) =$$

gradient at w

Informal sketch



By Taylor expansion around $\mathbf{w}^*$:

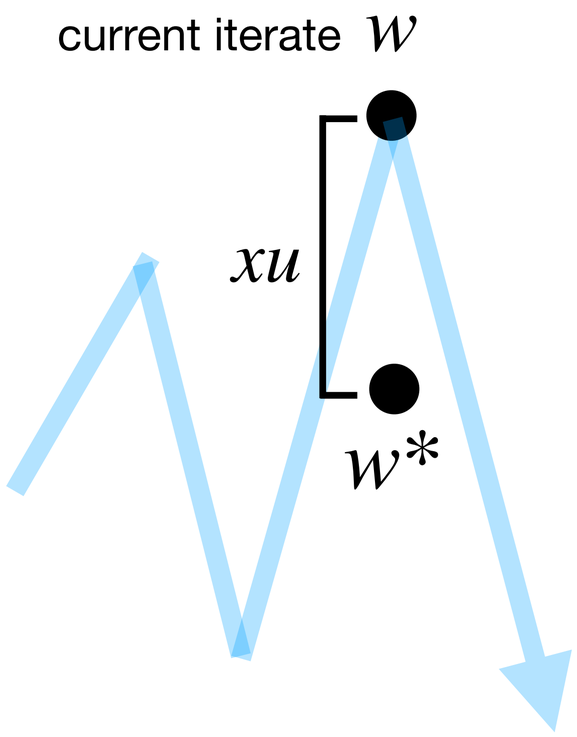
$$\nabla L(\mathbf{w}^* + x\mathbf{u}) = \nabla L(\mathbf{w}^*) + O(x)$$

gradient at w gradient at w^*

Informal sketch

Since u is a Hessian eigenvector

$$H(\mathbf{w}^*)\,\mathbf{u} = S(\mathbf{w}^*)\mathbf{u}$$



By Taylor expansion around $\mathbf{w}^*$:

$$\nabla L(\mathbf{w}^* + x\mathbf{u}) = \nabla L(\mathbf{w}^*) + \overbrace{H(\mathbf{w}^*)[x\mathbf{u}]}^{= S(\mathbf{w}^*)\,x\,\mathbf{u}} + O(x^2)$$

gradient at w — gradient at w^* — oscillation

- This term sends a negative gradient step computed at $\mathbf{w}^* + x\mathbf{u}$ towards the $-\mathbf{u}$ direction.
- This term is causing us to oscillate
- The “magic” comes from the *next* term in the Taylor expansion…

Informal sketch

- The next term in the Taylor expansion is:

curvature in u direction = $S(w^*)$

$$\nabla L(\mathbf{w}^* + x\mathbf{u}) = \nabla L(\mathbf{w}^*) + H(\mathbf{w}^*)[x\mathbf{u}] + \frac{1}{2}\, x^2 \nabla_{w^*}[\mathbf{u}^\top H(\mathbf{w}^*)\mathbf{u}] + O(x^3)$$

gradient at w | gradient at w^* | oscillation

gradient of curvature in u direction = $\nabla S(w^*)$

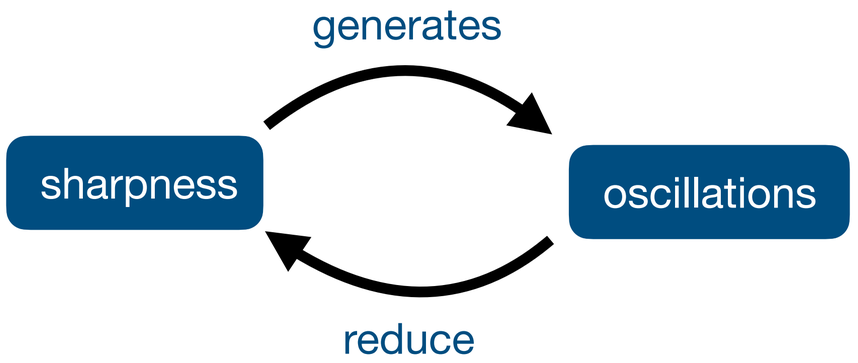
Informal sketch

- The next term in the Taylor expansion is:

$$\nabla L(\mathbf{w}^* + x\mathbf{u}) = \nabla L(\mathbf{w}^*) + H(\mathbf{w}^*)[x\mathbf{u}] + \frac{1}{2}x^2 \nabla S(\mathbf{w}^*) + O(x^3)$$

gradient at w | gradient at w^* | oscillation | gradient of sharpness

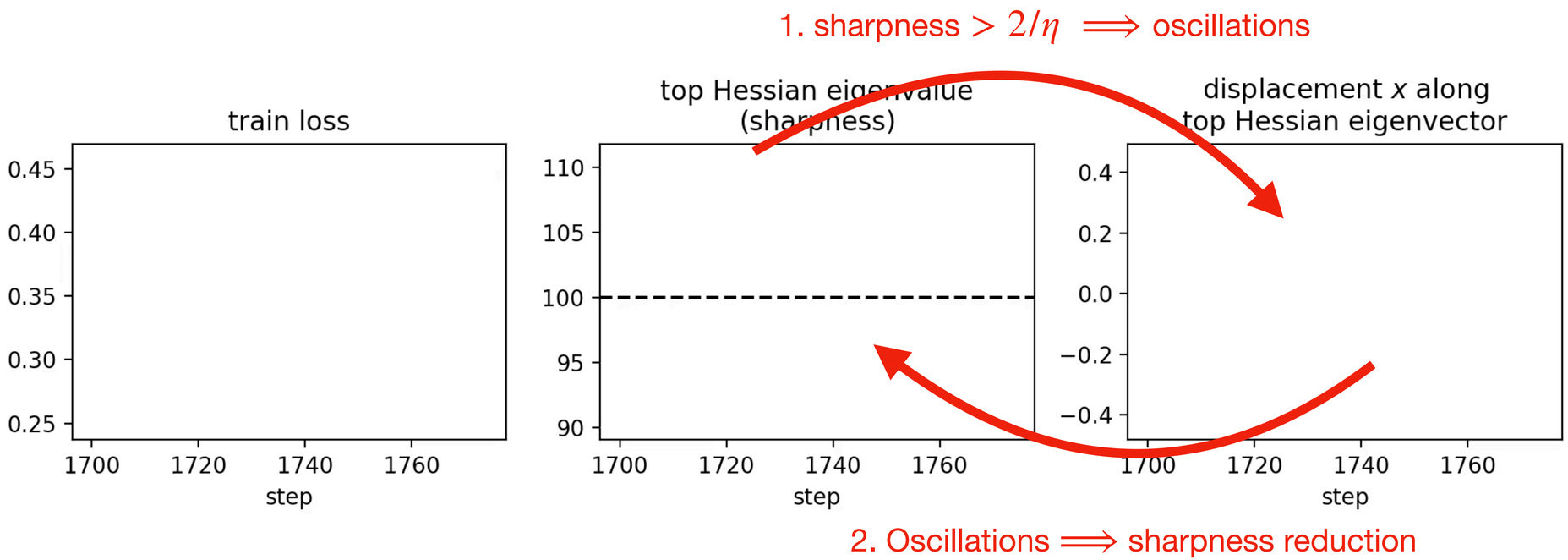
- Thus, a negative gradient step computed at $\mathbf{w}^* + x\mathbf{u}$ automatically takes a negative gradient step *on the sharpness* with step size $\frac{1}{2}\eta x^2$



Negative feedback loop!

Revisiting the behavior of GD

- Now we can finally understand the dynamics of gradient descent:



Revisiting the traditional theory

- Traditional theory doesn’t properly capture cause and effect
- Gradient descent doesn’t converge because the curvature is small…
  - … it converges because it *keeps* the curvature small
- Key ingredient: third-order Taylor expansion

Discussion

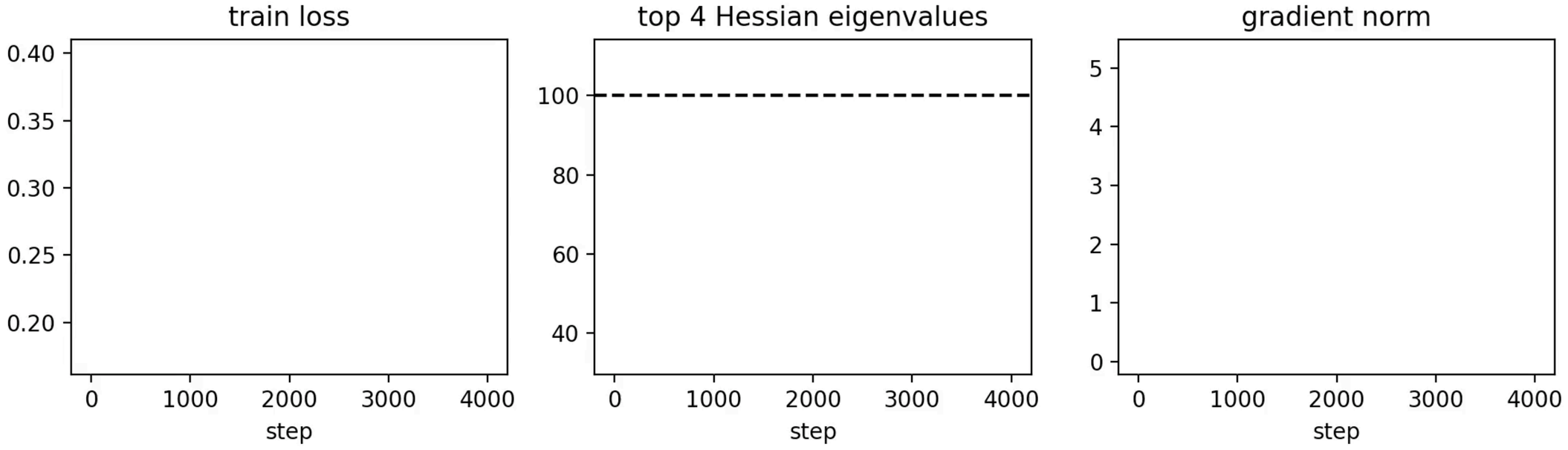
- I haven't used any properties of deep learning — only local Taylor expansions
  - We believe "self-stabilization" is a very generic behavior of GD, beyond DL
  - It is "progressive sharpening" that is the signature property of DL
- Precise claim: *if you start* stable, you remain stable
- See Alex & Eshaan's "self-stabilization" paper for a proof under certain assumptions
- The third-order story yields checkable numerical predictions (as you'll see)

How can we analyze gradient descent?

- Unfortunately, EOS dynamics are challenging to analyze in fine-grained detail
- Need to track the mutual interactions between oscillations and curvature
- There are frequently *multiple* unstable eigenvalues → chaotic dynamics

How can we analyze gradient descent?

- Unfortunately, EOS dynamics are challenging to analyze in fine-grained detail
- Need to track the mutual interactions between oscillations and curvature
- There are frequently *multiple* unstable eigenvalues → chaotic dynamics



How can we analyze gradient descent?

Cohen*, Damian*, Talwalkar, Kolter, Lee. *Understanding Optimization in Deep Learning with Central Flows.* ICLR '25.

Alex Damian

- We argue that the exact oscillatory GD trajectory doesn't matter
- Rather, what matters is the *macroscopic* (long-term) path that GD takes
- To characterize this path, we derive a differential equation called a *central flow*
- Our analysis is non-rigorous, but we show that it makes accurate *numerical* predictions on "real" neural nets

What path does gradient descent take?

- The usual continuous-time approximation to gradient descent is gradient flow:

$$\frac{dw}{dt} = -\eta \nabla L(w)$$

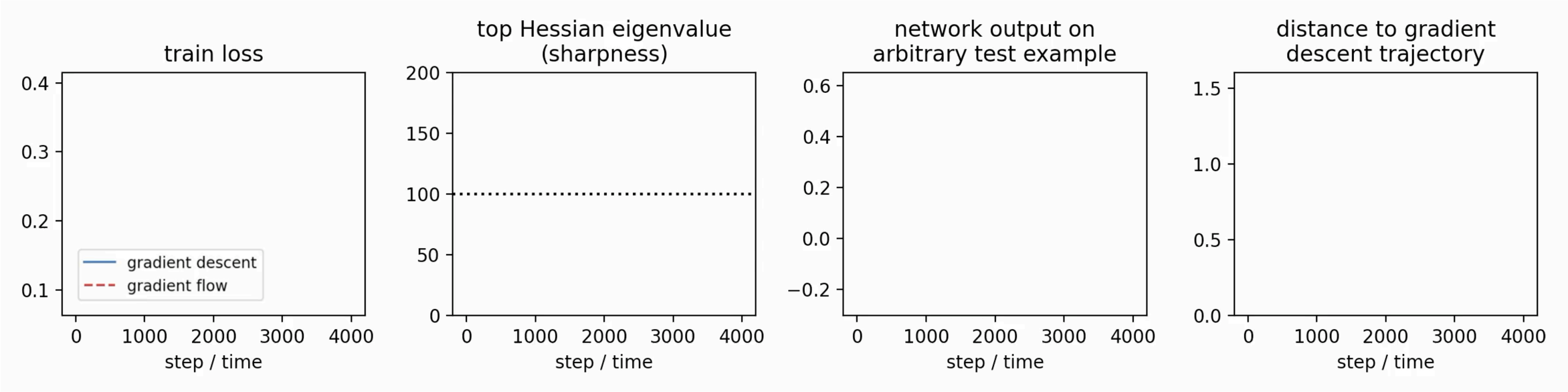
- Empirically, GD follows gradient flow *before* reaching EOS, but splits afterwards

What path does gradient descent take?

- The usual continuous-time approximation to gradient descent is gradient flow:

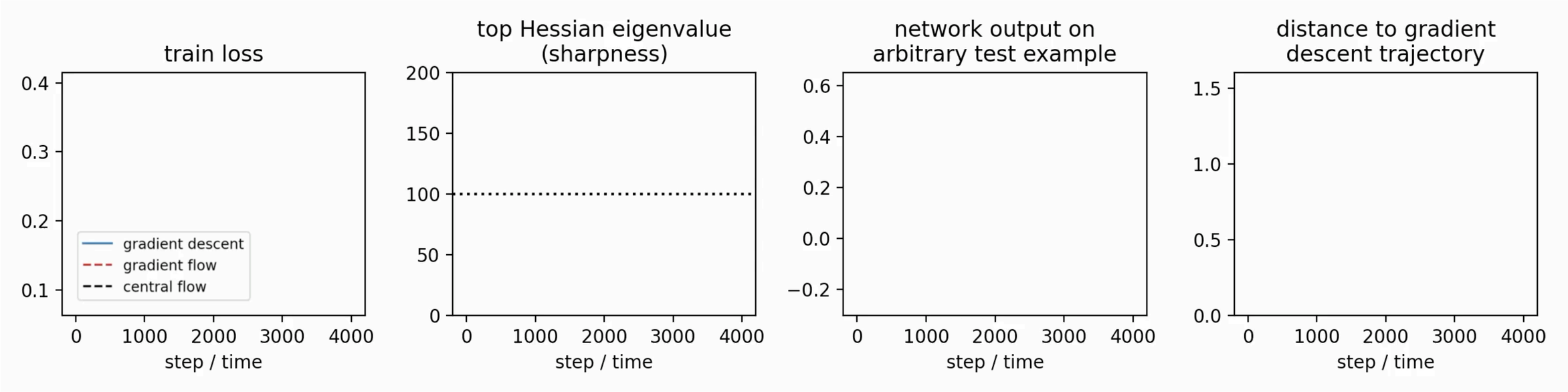
$$\frac{dw}{dt} = -\eta \nabla L(w)$$

- Empirically, GD follows gradient flow *before* reaching EOS, but splits afterwards



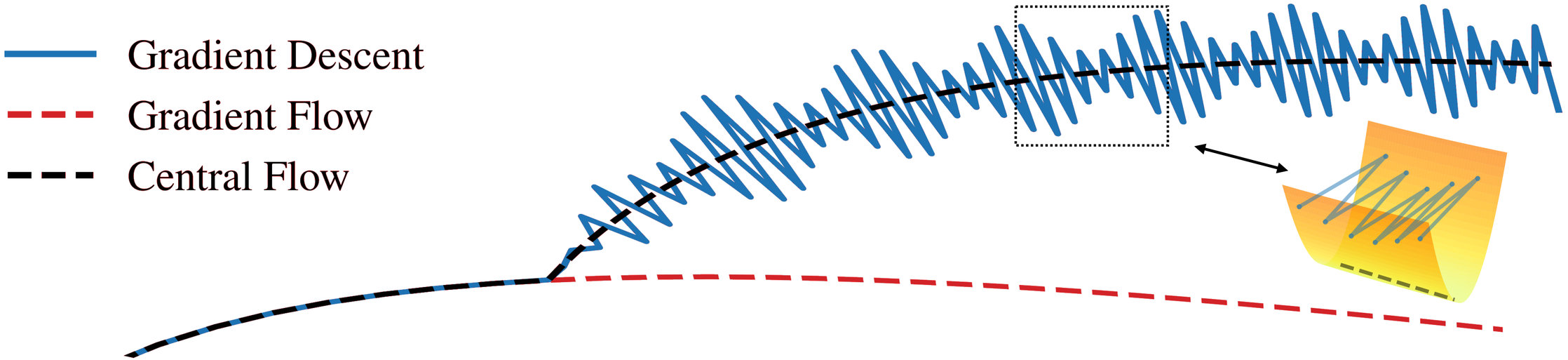
What path does gradient descent take?

- Our *central flow* matches the trajectory of gradient descent even at EOS:



The main idea, in one picture

- The central flow models the *time-averaged* (i.e. locally smoothed) GD trajectory



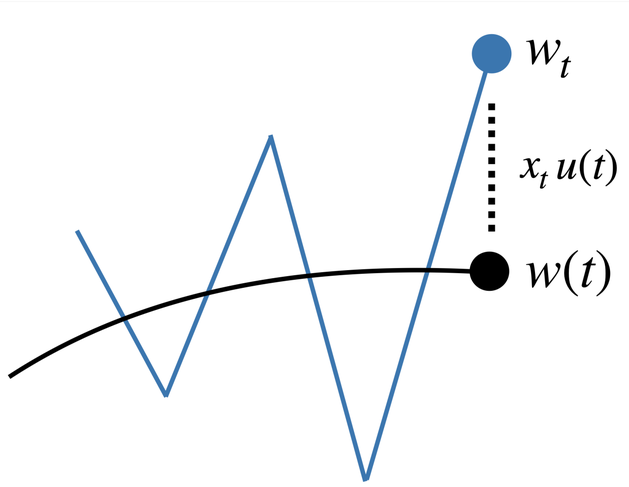
Central flow (one unstable eigenvalue)

CAUTION

Not rigorous!

We model the gradient descent trajectory as: $\mathbf{w}_t = \mathbf{w}(t) + x_t\,\mathbf{u}(t)$

iterate · central flow · oscillation magnitude · top Hessian eigenvector



By Taylor expansion, the gradient at $\mathbf{w}_t$ is:

gradient at iterate · gradient at flow · oscillation · gradient of sharpness

$$\nabla L(\mathbf{w}_t) = \nabla L(\mathbf{w}(t)) + x_t\,S(\mathbf{w}(t)) + \frac{1}{2}x_t^2\,\nabla S(\mathbf{w}(t))$$

Therefore, the "time-averaged" gradient is:

gradient at iterate · gradient at flow · implicit sharpness penalty

$$\mathbb{E}\,\nabla L(\mathbf{w}_t) = \nabla L(\mathbf{w}(t)) + \cancel{x_t\,S(\mathbf{w}(t))\mathbf{u}(t)} + \frac{1}{2}\mathbb{E}[x_t^2]\,\nabla S(\mathbf{w}(t))$$

variance of oscillations

Central flow (one unstable eigenvalue)

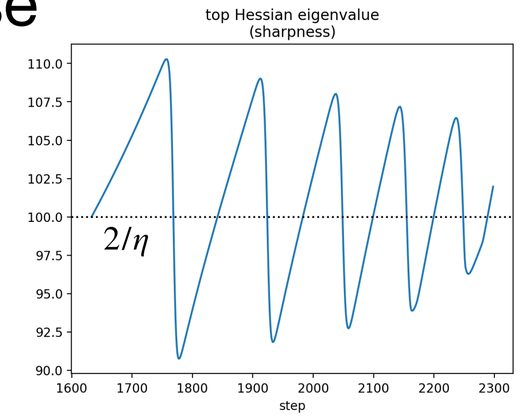
- We suppose that the time-averaged GD trajectory follows an ODE of the form:

"instantaneous variance" of the oscillations
(i.e. local time average of x^2)

$$\frac{dw}{dt} = -\eta \left[\nabla L(w) + \frac{1}{2}\sigma^2(t) \nabla S(w) \right]$$

gradient flow — sharpness penalty

- This flow averages out the oscillations, but keeps their *effect* on the trajectory.
- To determine $\sigma^2(t)$, we argue that only one value "makes sense"
  - Empirically, the sharpness equilibrates at $2/\eta$.
  - Therefore, we enforce that along the central flow, $\frac{dS}{dt} = 0$.



Central flow (one unstable eigenvalue)

- The time derivative of the sharpness under our flow is:

$$\frac{dS}{dt} = \langle \nabla S(w), \frac{dw}{dt} \rangle \quad \text{chain rule}$$

$$= \left\langle \nabla S(w), -\eta \left[\nabla L(w) + \tfrac{1}{2} \sigma^2(t) \nabla S(w) \right] \right\rangle \quad \text{substitute in our flow}$$

$$= \langle \nabla S(w), -\eta \nabla L(w) \rangle - \frac{1}{2} \eta \sigma^2(t) \|\nabla S(w)\|^2 \quad \text{simplify}$$

time derivative of sharpness under gradient flow

sharpness-reduction effect of oscillations

- We see that $\frac{dS}{dt}$ is **affine** in $\sigma^2(t)$. In order for $\frac{dS}{dt} = 0$, $\sigma^2(t)$ must be:

$$\sigma^2(t) = \frac{2 \langle -\nabla L(w), \nabla S(w) \rangle}{\|\nabla S(w)\|^2}$$

Central flow (one unstable eigenvalue)

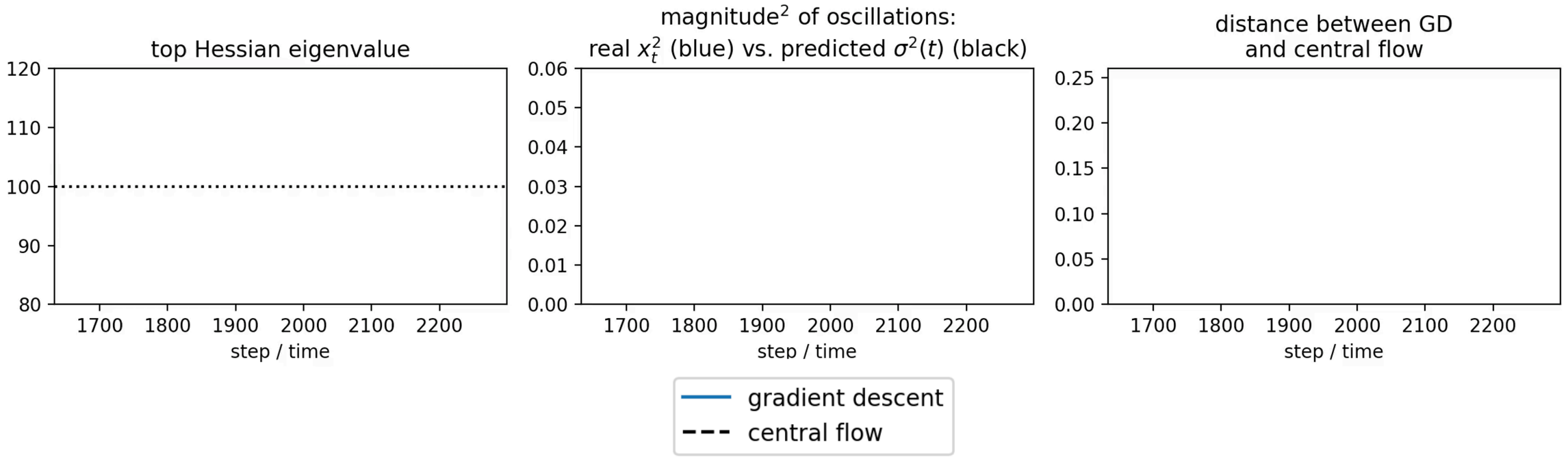
- The central flow for a single unstable eigenvalue is:

$$\frac{dw}{dt} = -\eta \left[\nabla L(w) \; + \; \frac{1}{2} \sigma^2(t) \, \nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2 \nabla L(w), \nabla S(w) \rangle}{\| \nabla S(w) \|^2}$$

Central flow (one unstable eigenvalue)

- The central flow for a single unstable eigenvalue is:

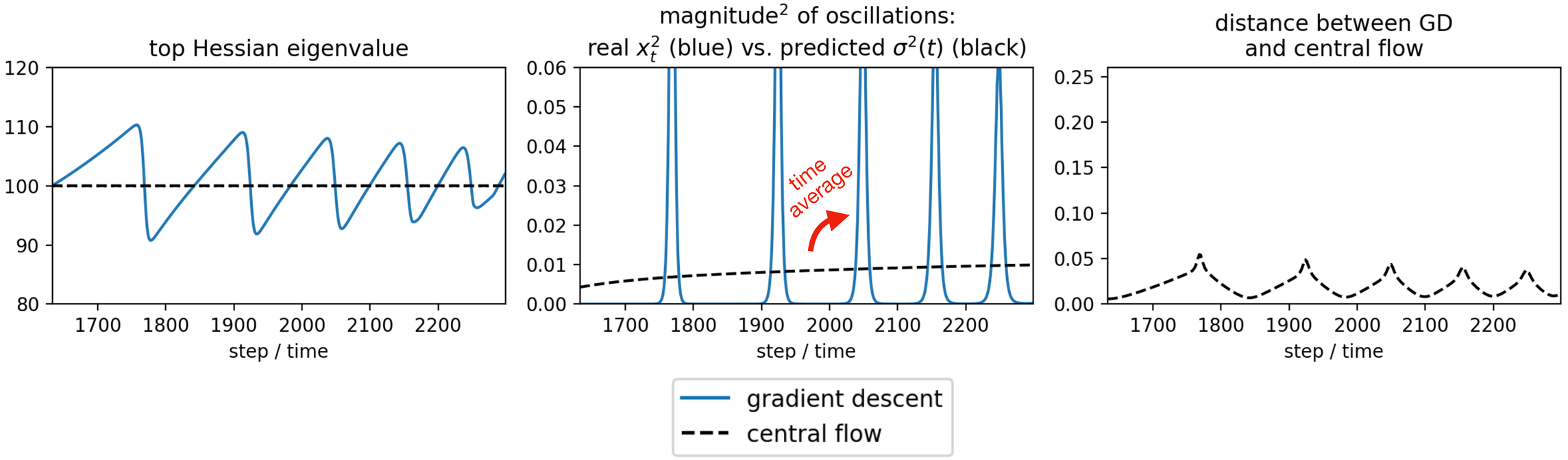
$$\frac{dw}{dt} = -\eta \left[\nabla L(w) \; + \; \frac{1}{2} \sigma^2(t) \, \nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2 \nabla L(w), \nabla S(w) \rangle}{\| \nabla S(w) \|^2}$$



Central flow (one unstable eigenvalue)

- The central flow for a single unstable eigenvalue is:

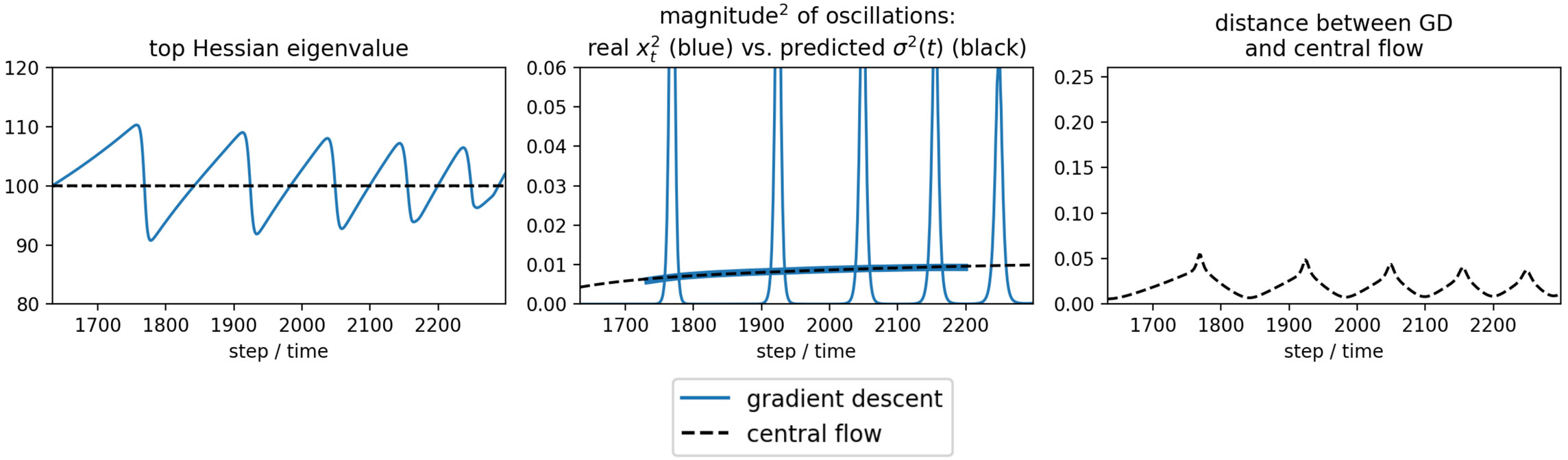
$$\frac{dw}{dt} = -\eta \left[\nabla L(w) \; + \; \frac{1}{2} \sigma^2(t) \, \nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2 \nabla L(w), \nabla S(w) \rangle}{\| \nabla S(w) \|^2}$$



Central flow (one unstable eigenvalue)

- The central flow for a single unstable eigenvalue is:

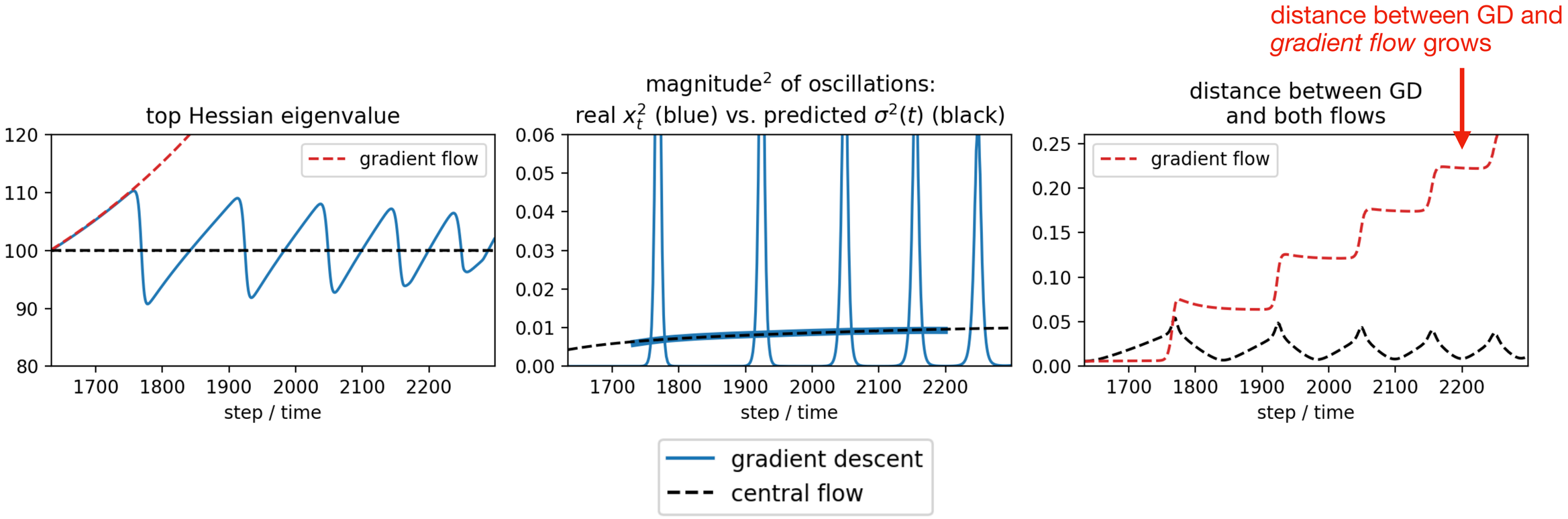
$$\frac{dw}{dt} = -\eta \left[\nabla L(w) \; + \; \frac{1}{2} \sigma^2(t) \, \nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2 \nabla L(w), \nabla S(w) \rangle}{\| \nabla S(w) \|^2}$$



Central flow (one unstable eigenvalue)

- The central flow for a single unstable eigenvalue is:

$$\frac{dw}{dt} = -\eta \left[\nabla L(w) \; + \; \frac{1}{2} \sigma^2(t) \, \nabla S(w) \right] \quad \text{where} \quad \sigma^2(t) = \frac{\langle -2 \nabla L(w), \nabla S(w) \rangle}{\| \nabla S(w) \|^2}$$

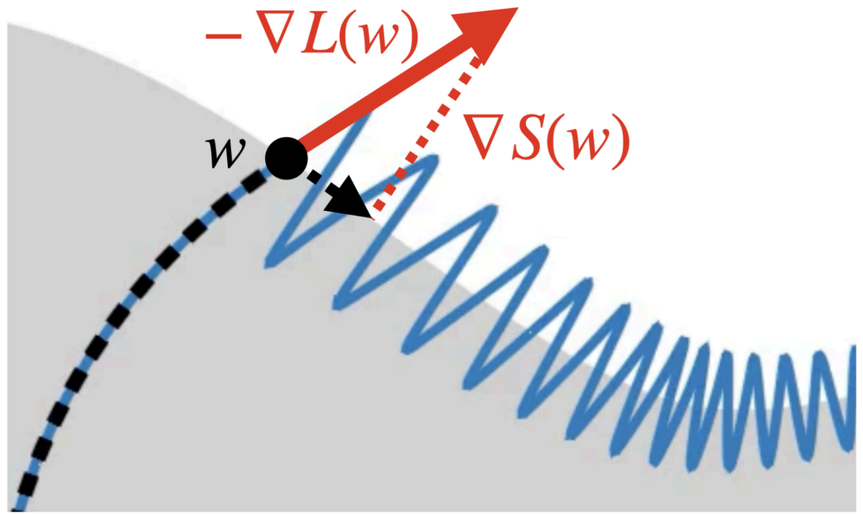


Interpretation as projection

- Substitute the expression for $\sigma^2(t)$ into the central flow:

$$\frac{dw}{dt} = -\eta \left[I - \frac{\nabla S(w) \nabla S(w)^\top}{\|\nabla S(w)\|^2} \right] \nabla L(w)$$

- "Project the $\nabla S(w)$ direction out from $\nabla L(w)$"
- Implies loss decreases monotonically



Complete central flow

CAUTION

Not rigorous!

We make the ansatz that GD follows a central flow of the form:

$$\frac{dw}{dt} = -\eta \left[\underbrace{\nabla L(w)}_{\text{gradient flow}} + \underbrace{\frac{1}{2} \nabla_w \langle H(w), \Sigma(t) \rangle}_{\text{implicit curvature penalty}} \right]$$

penalize "Σ-weighted Hessian"

- We similarly argue that only one value "makes sense" for $\Sigma(t)$.

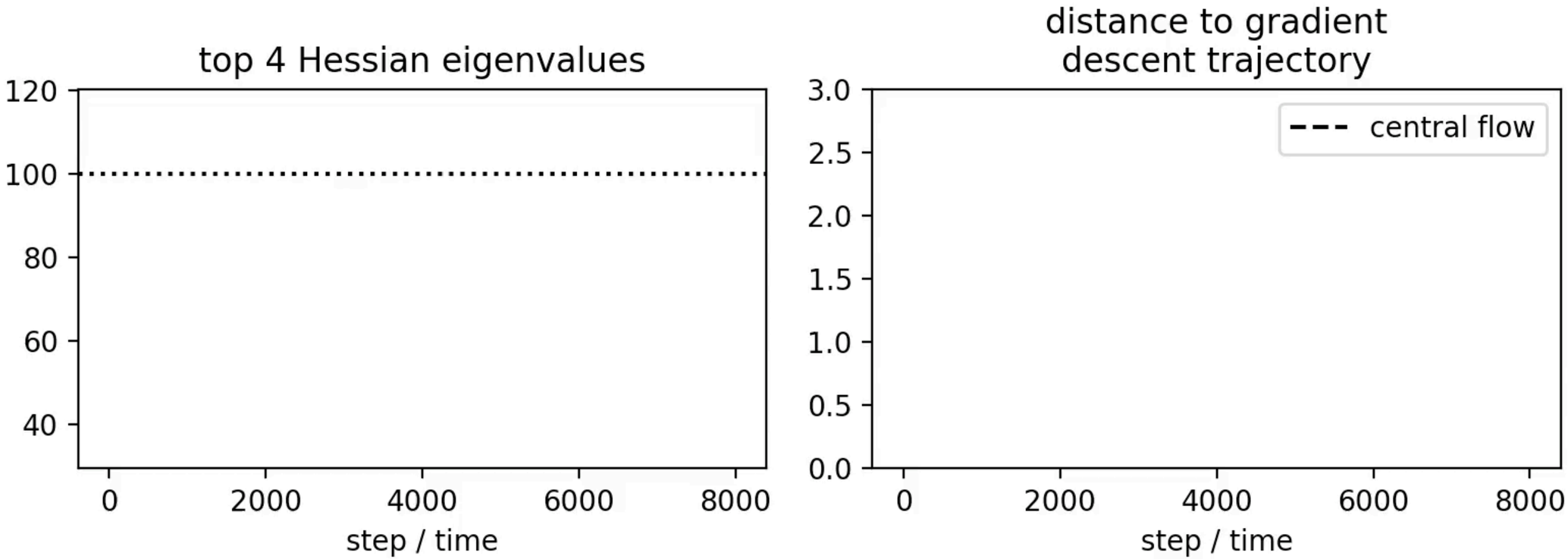
Complete central flow

- We impose three conditions:

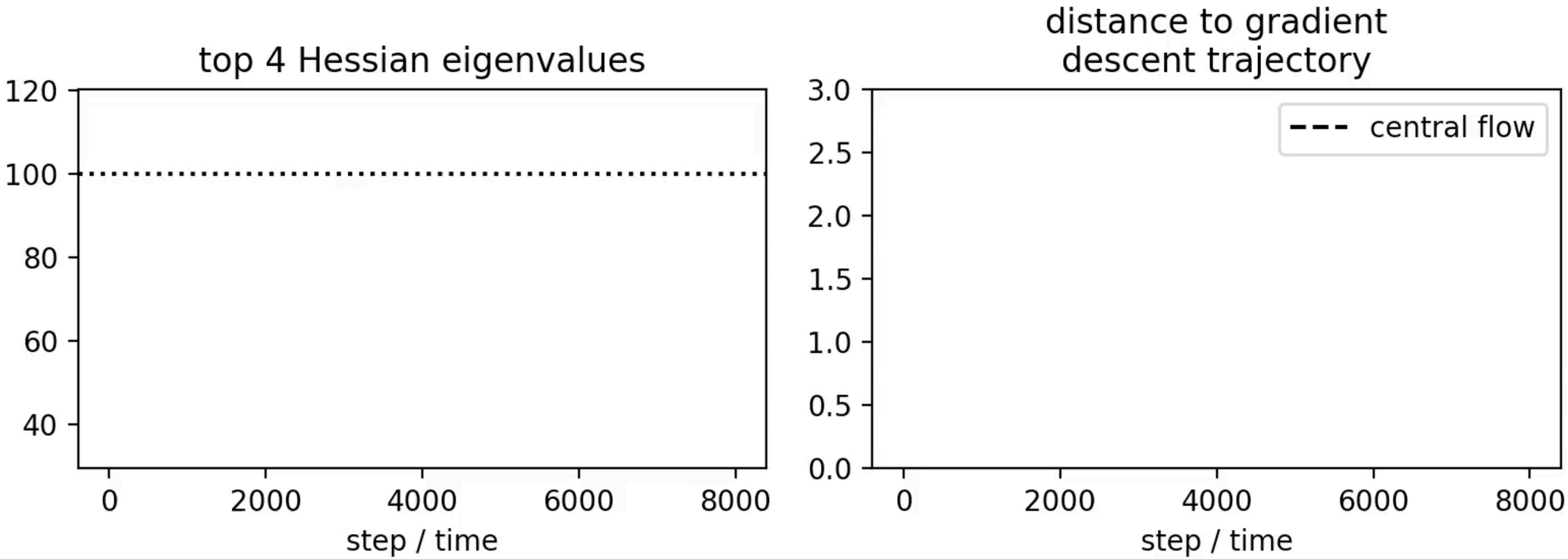
  1. The flow should not allow any Hessian eigenvalue to go above $2/\eta$
  2. $\Sigma(t)$ should be supported within the Hessian's $2/\eta$ eigenspace
  3. $\Sigma(t)$ should be positive semidefinite

- These three conditions imply that $\Sigma(t)$ must be the unique solution to a certain convex program called a *semidefinite complementarity problem.*
- The central flow is defined with this $\Sigma(t)$.
- Note: when sharpness $< 2/\eta$, $\Sigma(t) = 0$ and we get gradient flow.

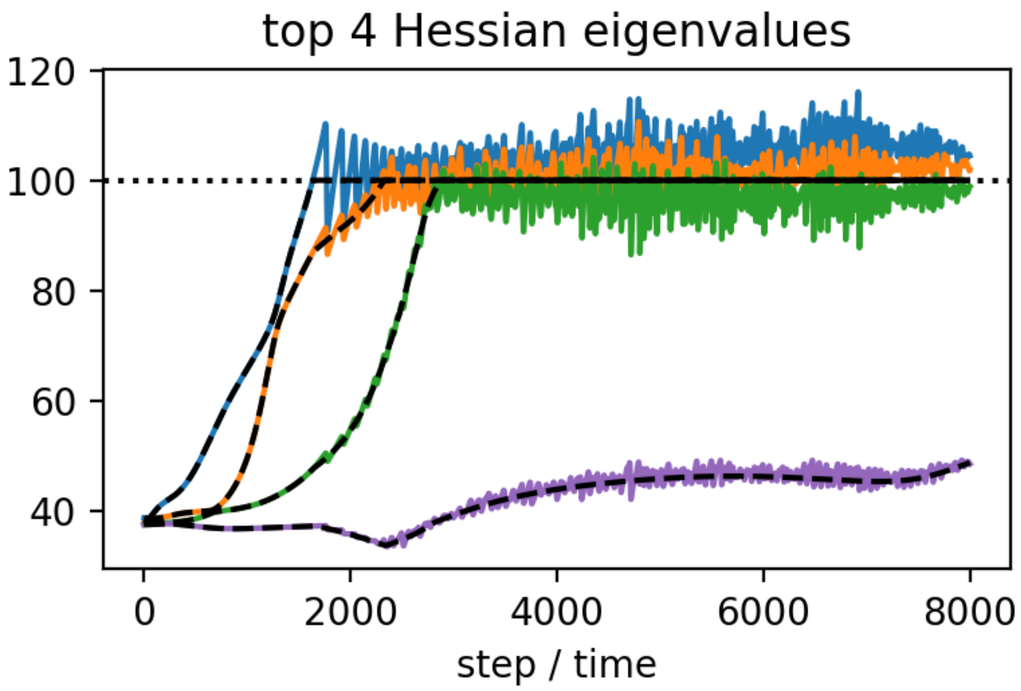
Central flow in action

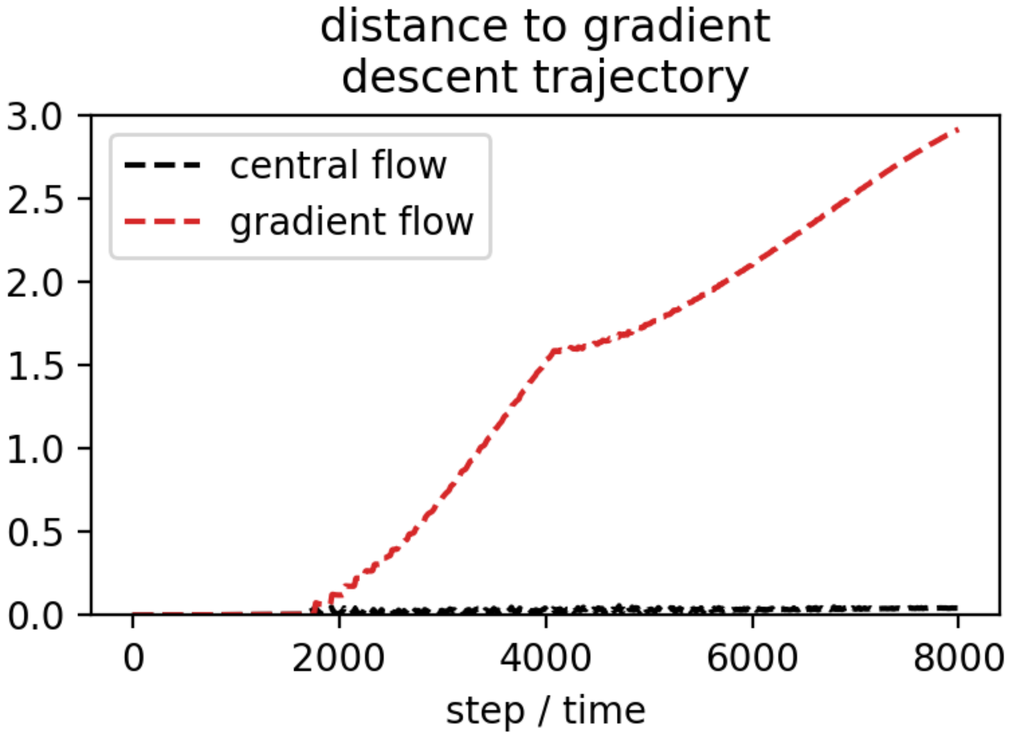


Central flow in action

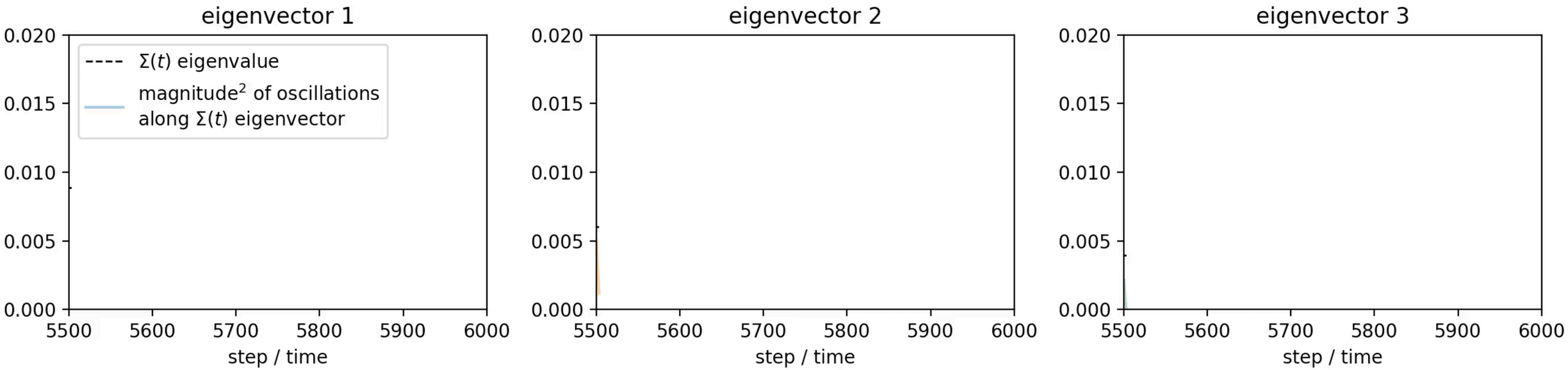


Central flow in action

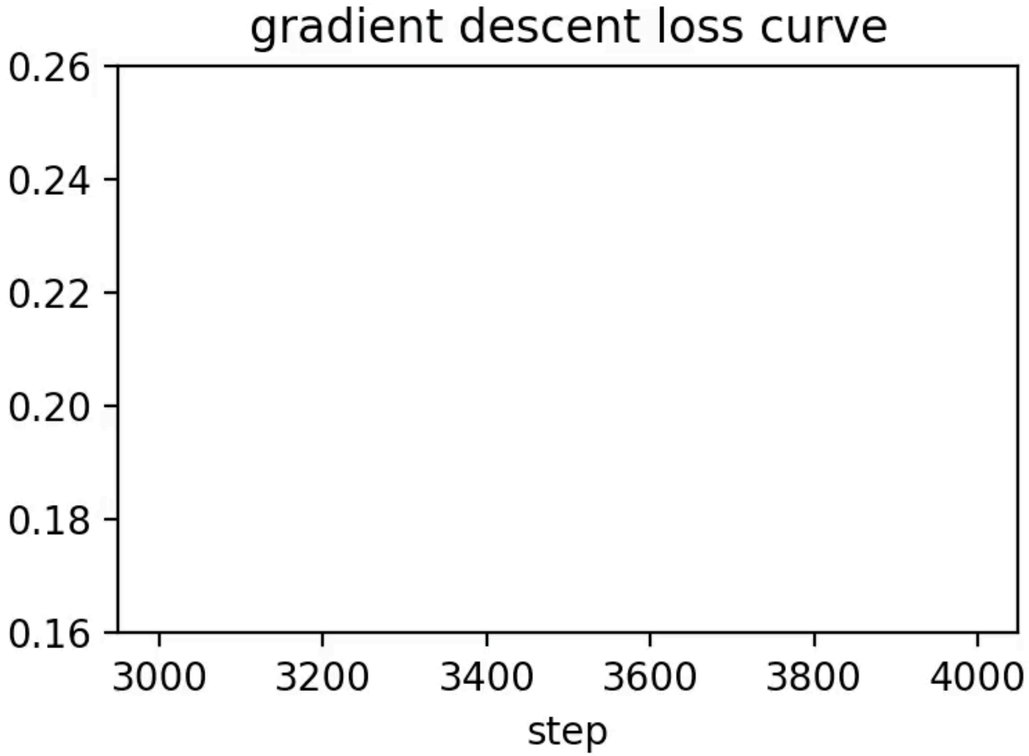




Σ accurately predicts oscillation covariance!

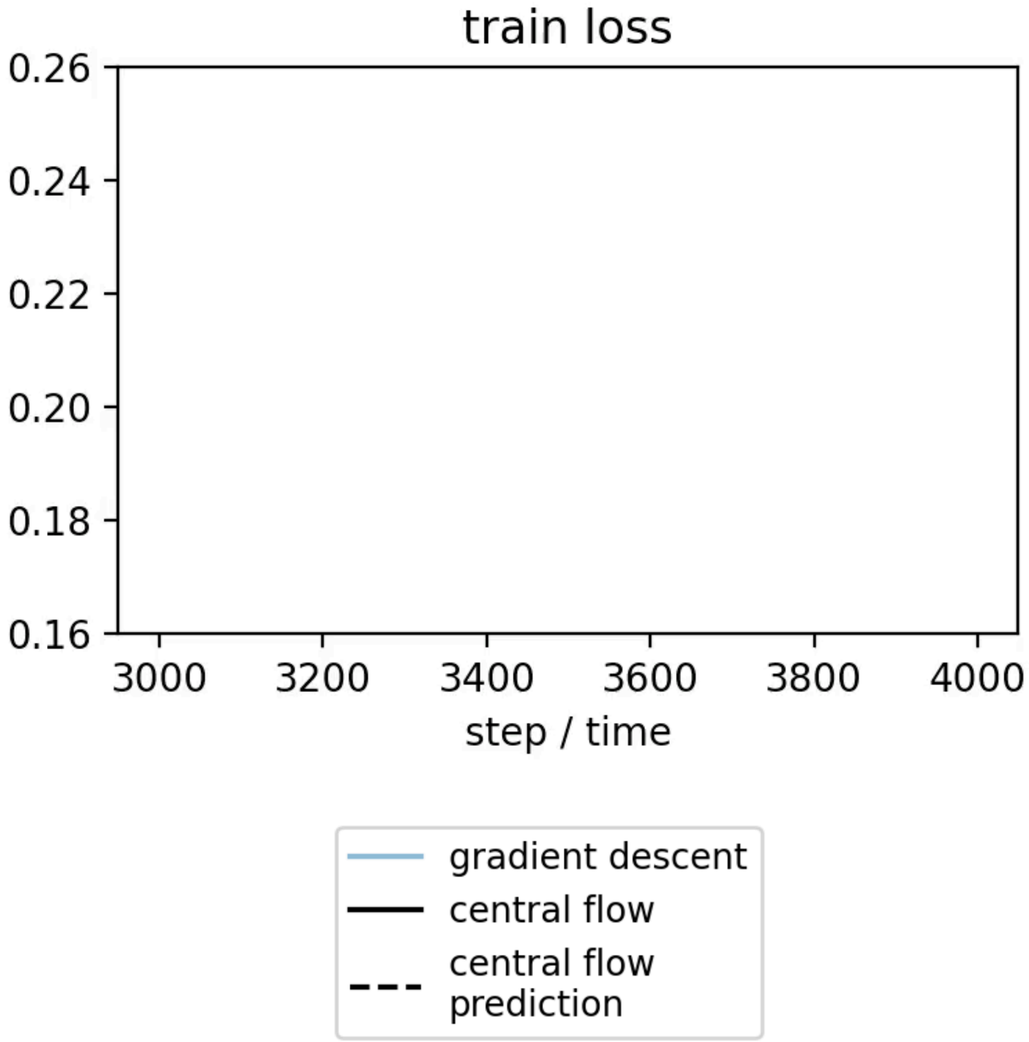


Reasoning about loss curves



- The train loss is non-monotonic over the short term, but decreases over the long term

Reasoning about loss curves

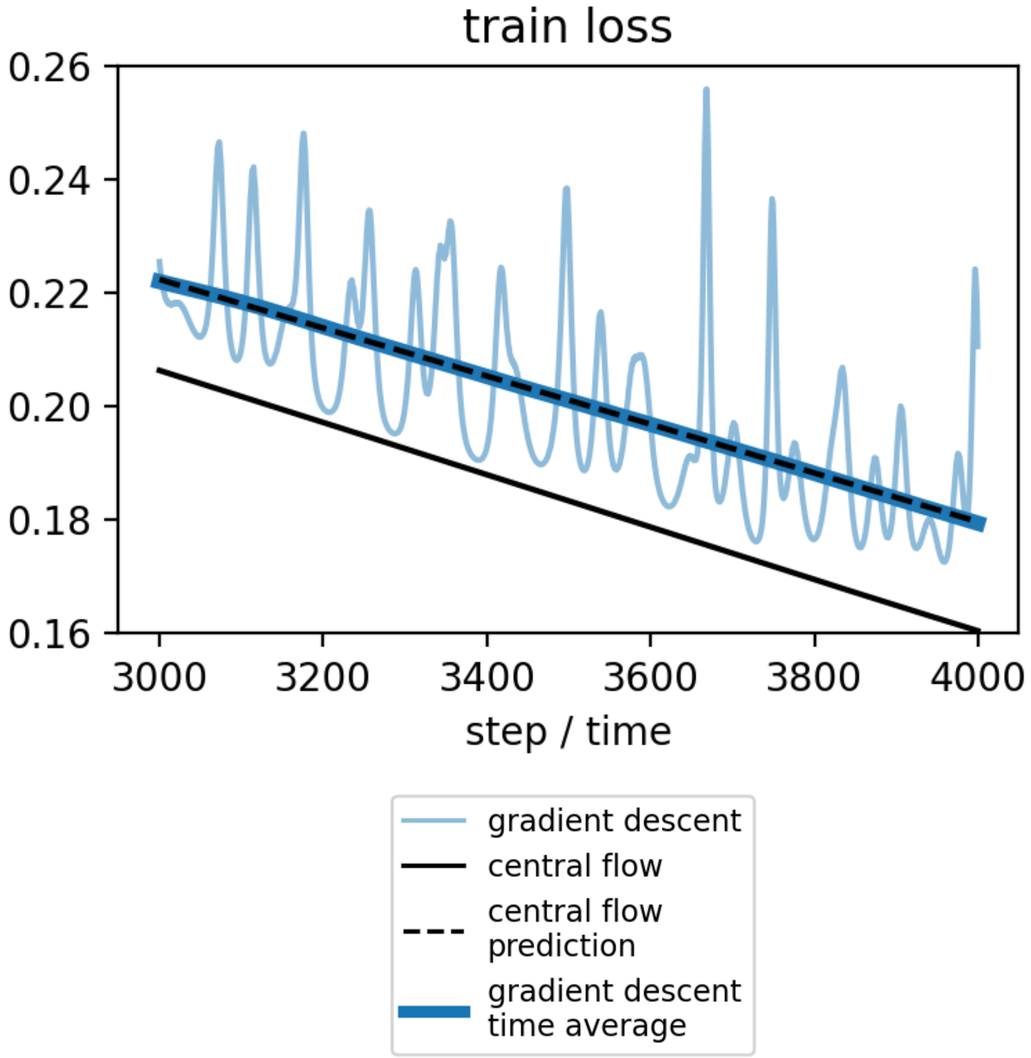


- Central flow lets us decompose the loss curve into a part that decreases monotonically, and a part from oscillations

$$\mathbb{E}[L(w_t)] \approx L(w(t)) + \frac{1}{\eta} \operatorname{tr} \Sigma(t)$$

time-averaged GD loss | loss along central flow | contribution from oscillations

Reasoning about loss curves

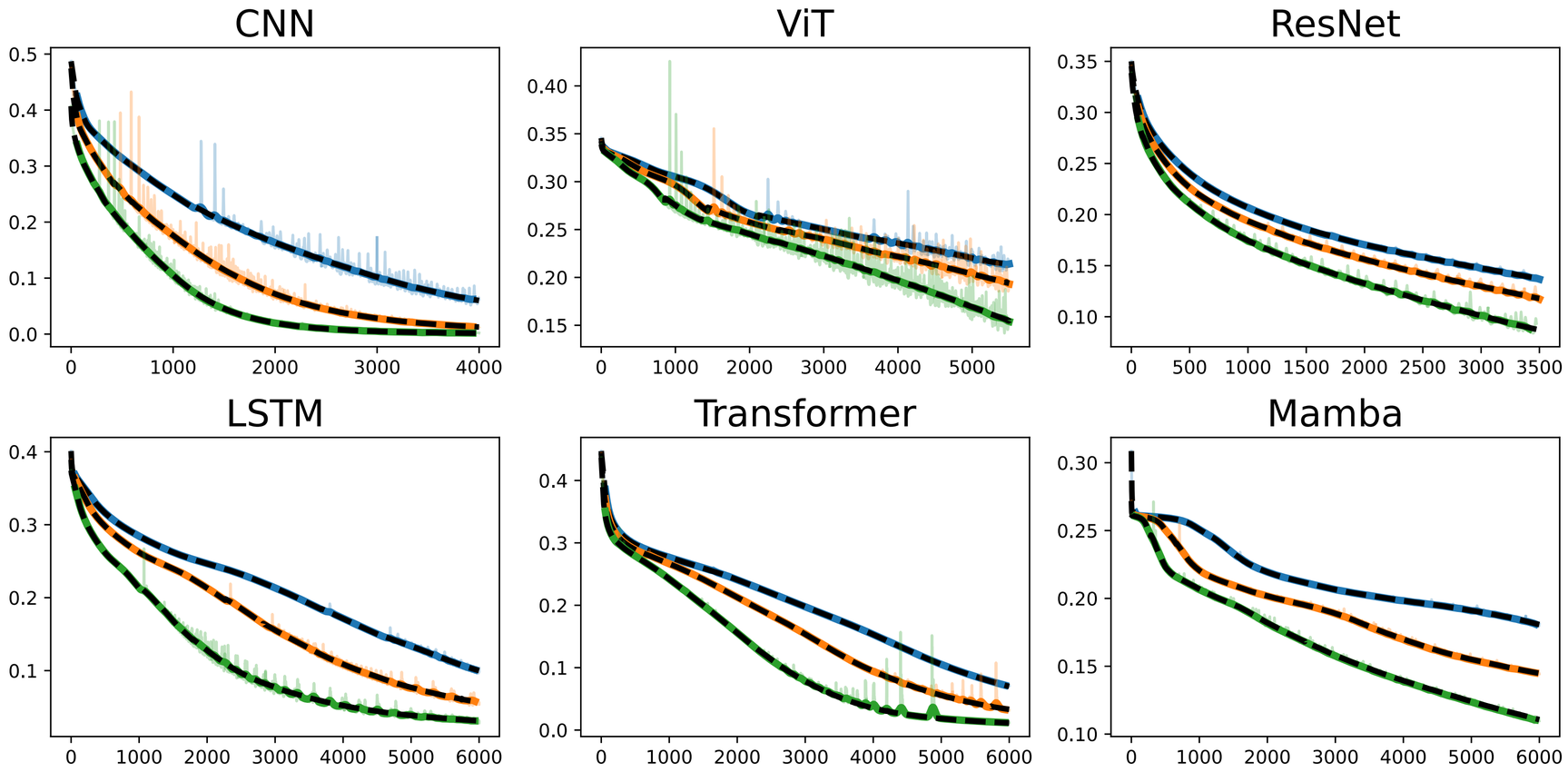


- Central flow lets us decompose the loss curve into a part that decreases monotonically, and a part from oscillations

$$\mathbb{E}[L(w_t)] \approx L(w(t)) + \frac{1}{\eta} \operatorname{tr} \Sigma(t)$$

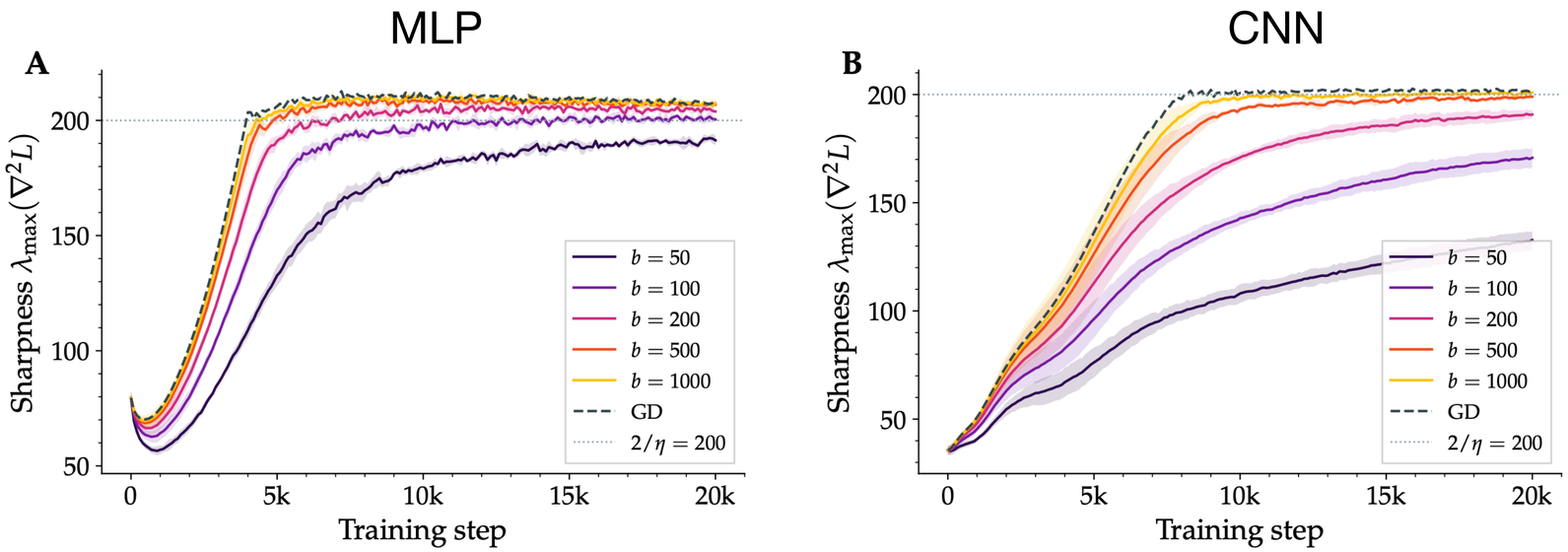
time-averaged GD loss | loss along central flow | contribution from oscillations

Our analysis applies to generic neural nets



What about *stochastic* gradient descent?

- SGD oscillations/fluctuations also induce implicit curvature penalty
  - But SGD fluctuates *before* $2/\eta$, so sharpness is regulated before reaching $2/\eta$



Liao, Kolomvaki, Kyrillidis. "SGD at the Edge of Stability: The Stochastic Sharpness Gap." arXiv 2026.

- Broader point stands: L-smoothness is dynamically regulated by algorithm

“So what? Does this suggest a better algorithm?”

- That’s not our immediate goal
- Practical algorithm design is currently bottlenecked on understanding
- The way to maximize our *long-term* impact on practice is to focus on understanding the foundations now
- Maybe someone will put this theory to good use next month
  - Or maybe it will take years
- We have faith that eventually, a true understanding will yield practical gains
  - Maybe big ones …

Paper and blog site

- Paper: “Understanding Optimization in Deep Learning with Central Flows”
- Blog site: http://centralflows.github.io
  - Part 1: How does gradient descent work?
  - Part 2: A simple adaptive optimizer
  - Part 3: How does RMSProp work?