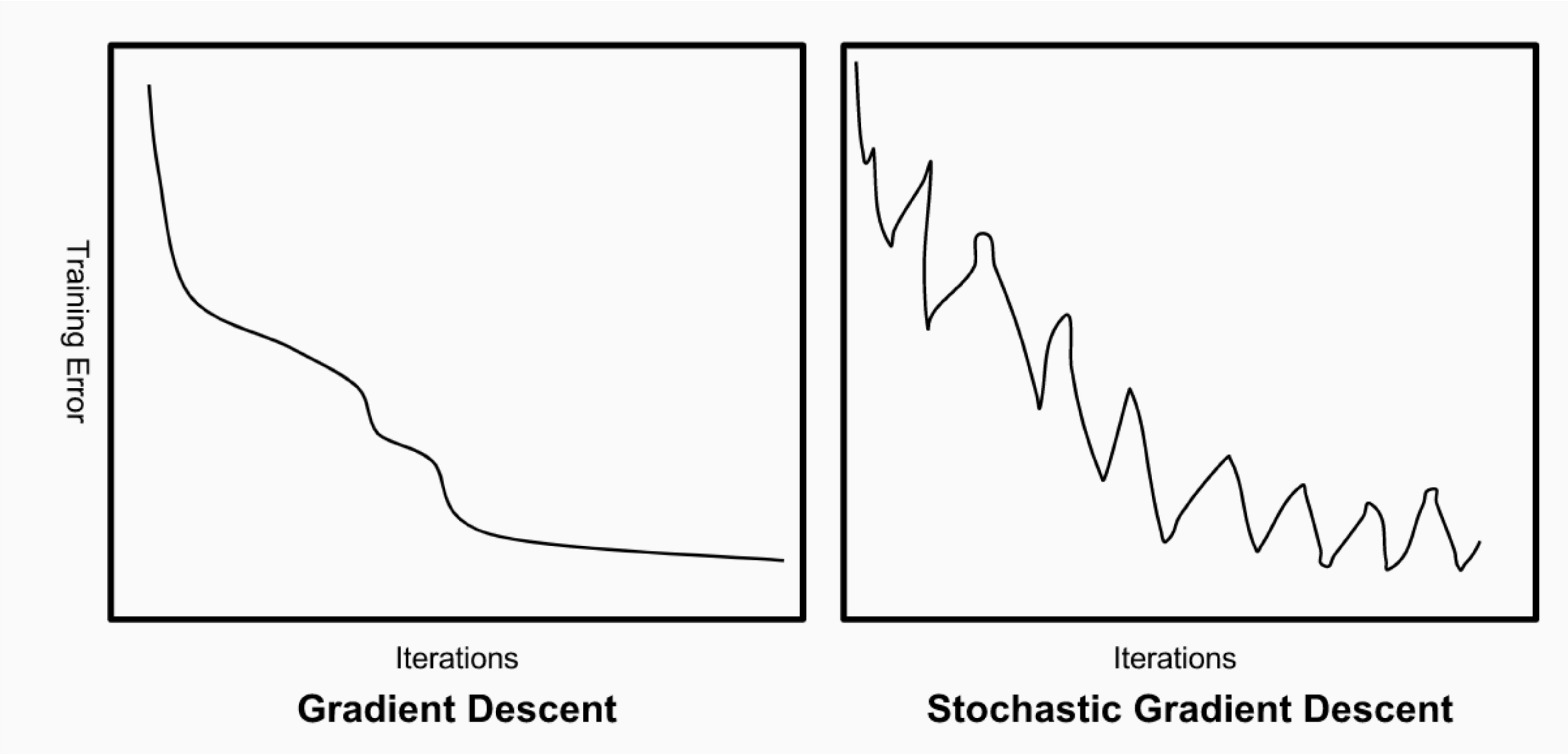
Gradient Descent on Neural Networks Typically Occurs on the Edge of Stability

Authors: Jeremy Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, Ameet Talwalkar

Presented by: Honghua Dong and Tianxing Li

Should full-batch gradient descent decrease training loss monotonically?



Source: http://pages.cs.wisc.edu/~spehlmann/cs760/_site//project/2017/05/04/intro.html

Stability of Gradient Descent on Quadratics

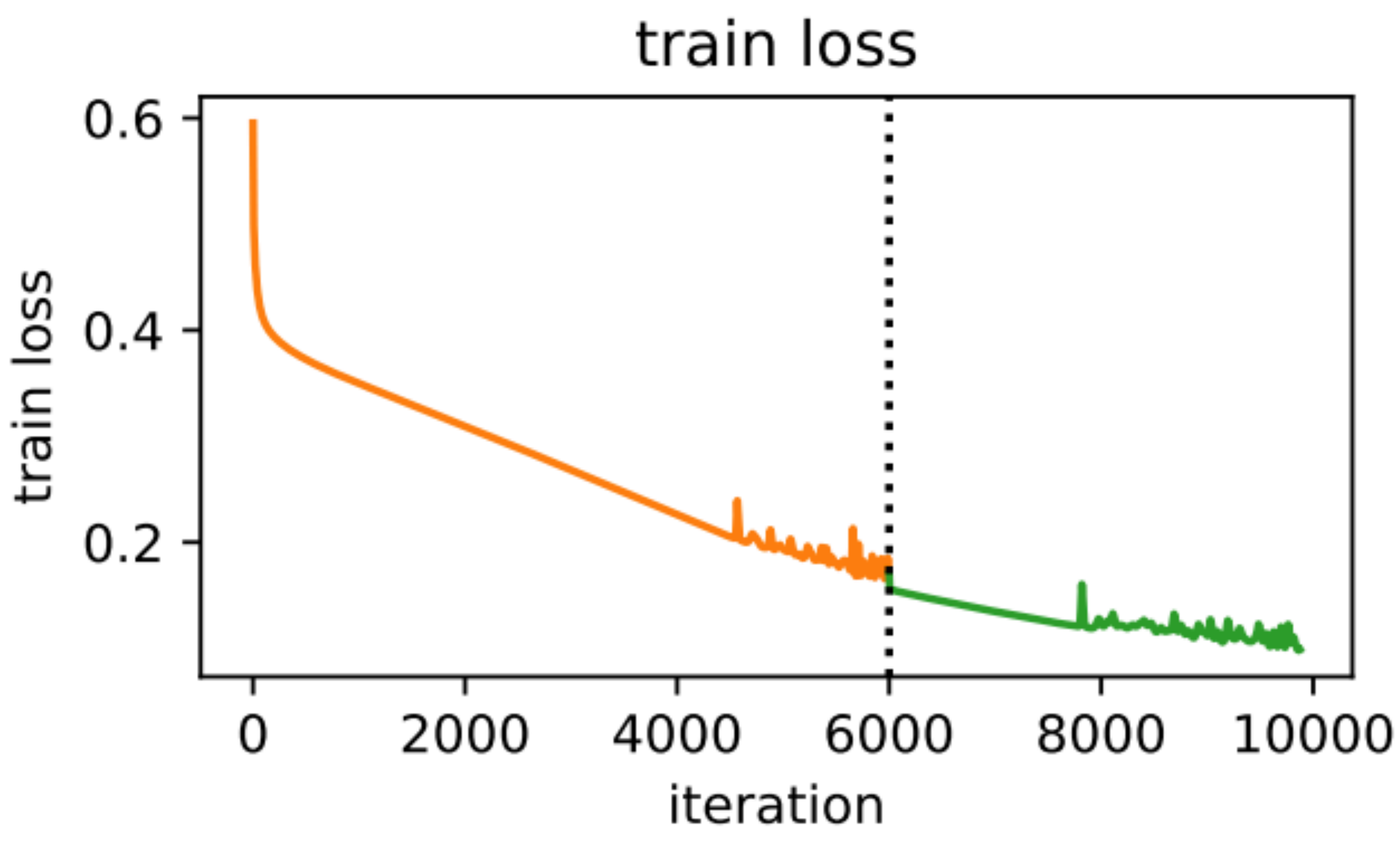
- *Objective:* $f(x) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$
- Coordinates evolve independently along eigenvectors of $\mathbf{A}$.
- *GD Update:* $x_{t+1} = x_t - \eta(ax_t + b)$
- *Optimum:* $x^* = -b/a$
- *Dynamics:* $x_t = (1 - \eta a)^t(x_0 - x^*) + x^*$
- *Stability Criterion (assuming $a \geq 0$):* $a \leq 2/\eta$

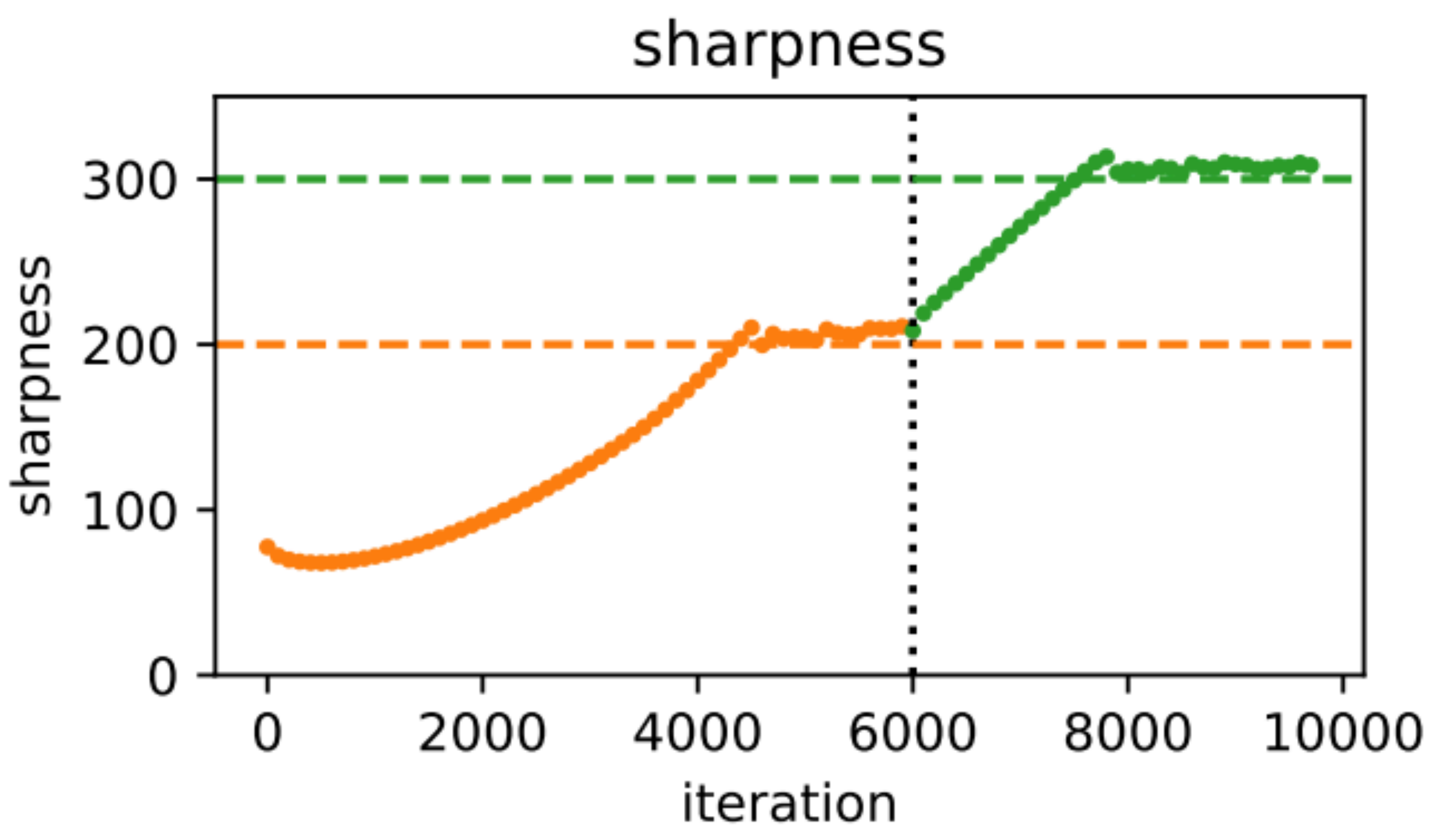
Local Quadratic Approximation to NNs

- $\mathscr{L}(\mathbf{x}) \approx \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T \mathbf{H}(\mathbf{x} - \mathbf{x}_0) + \mathbf{J}(\mathbf{x} - \mathbf{x}_0) + \mathscr{L}(\mathbf{x}_0)$
- If we used GD on this approximation, coordinates evolve independently along eigenvectors of $\mathbf{H}$.
- GD diverges along eigenvectors with negative curvature.
- GD diverges along eigenvectors with curvature $h > 2/\eta$

The Edge of Stability

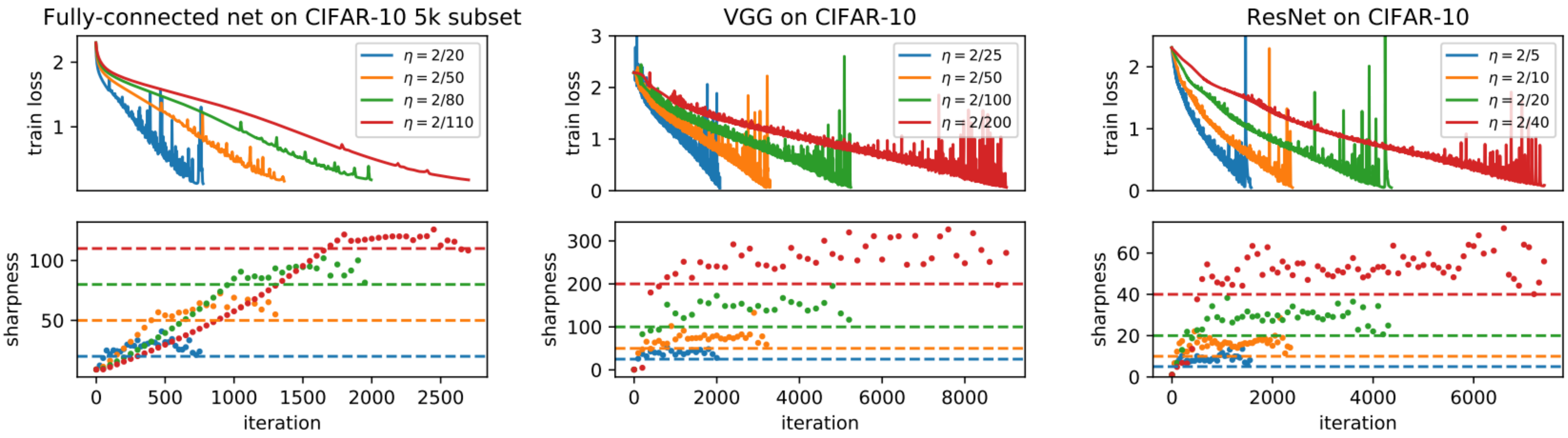
- "Sharpness": h_{max}, the maximum eigenvalue of $\mathbf{H}$
- **Sharpness increases to $2/\eta$, then hovers above $2/\eta$.**
- If we drop the learning rate, the sharpness will adjust itself!





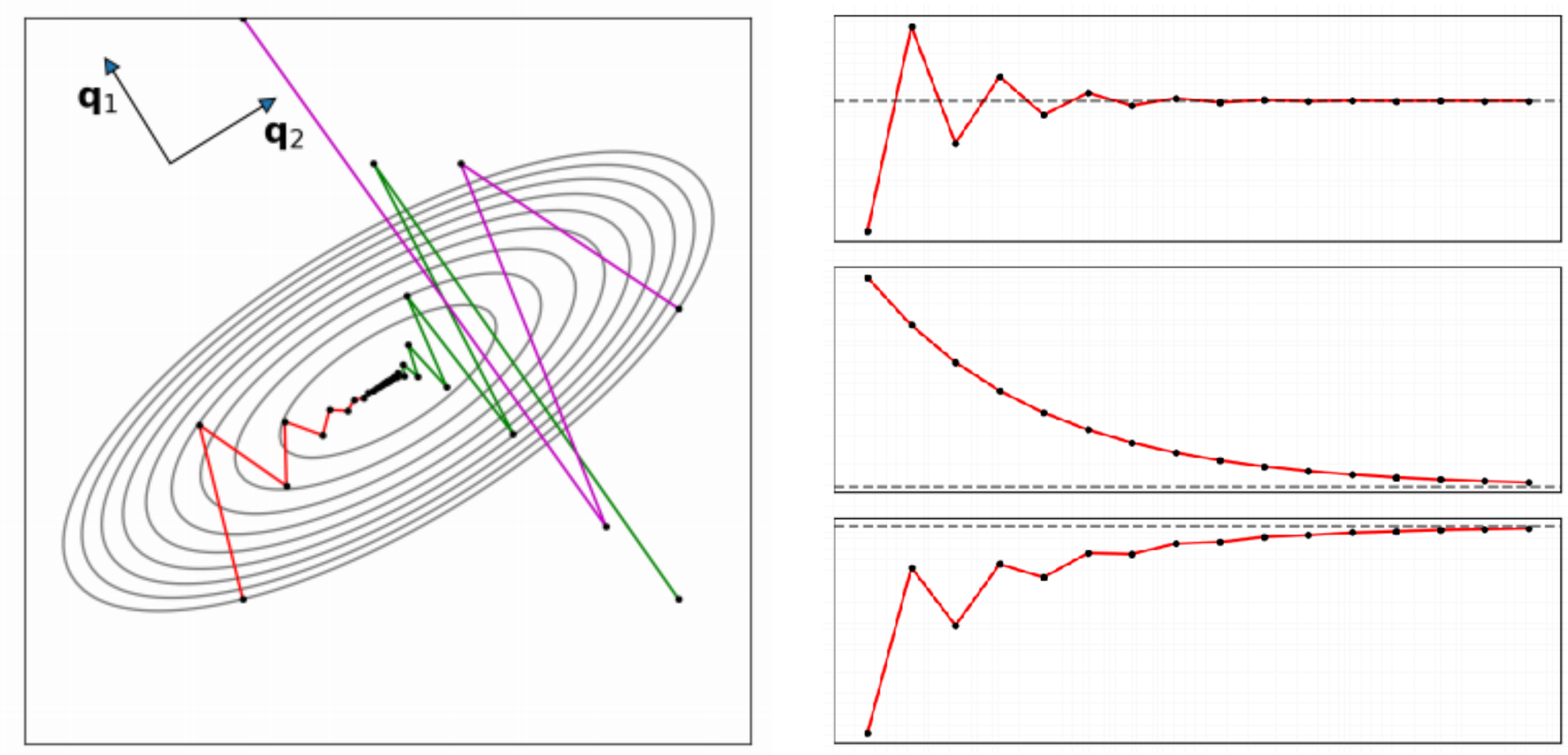
The Edge of Stability

- “Edge of Stability”: the regime under which the sharpness hovers near $2/\eta$



A Closer Look

- Compute leading eigenvector of $\mathbf{H}$ at some point of training, call it $\mathbf{q}_1$
- We can project the weights along $\mathbf{q}_1$ to visualize the iterates

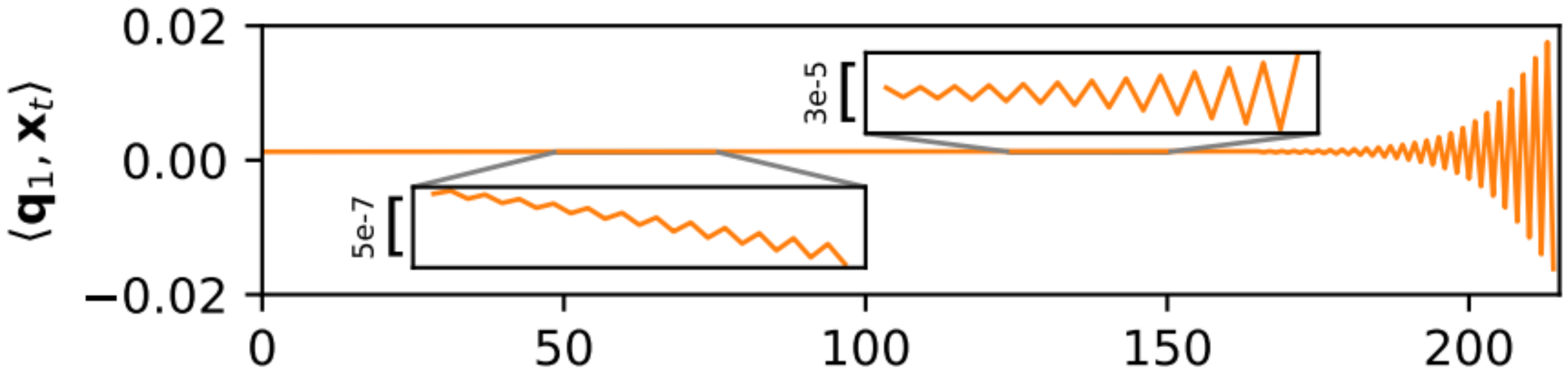


Source: https://www.cs.toronto.edu/~rgrosse/courses/csc2541_2021/readings/L01_intro.pdf

A Closer Look

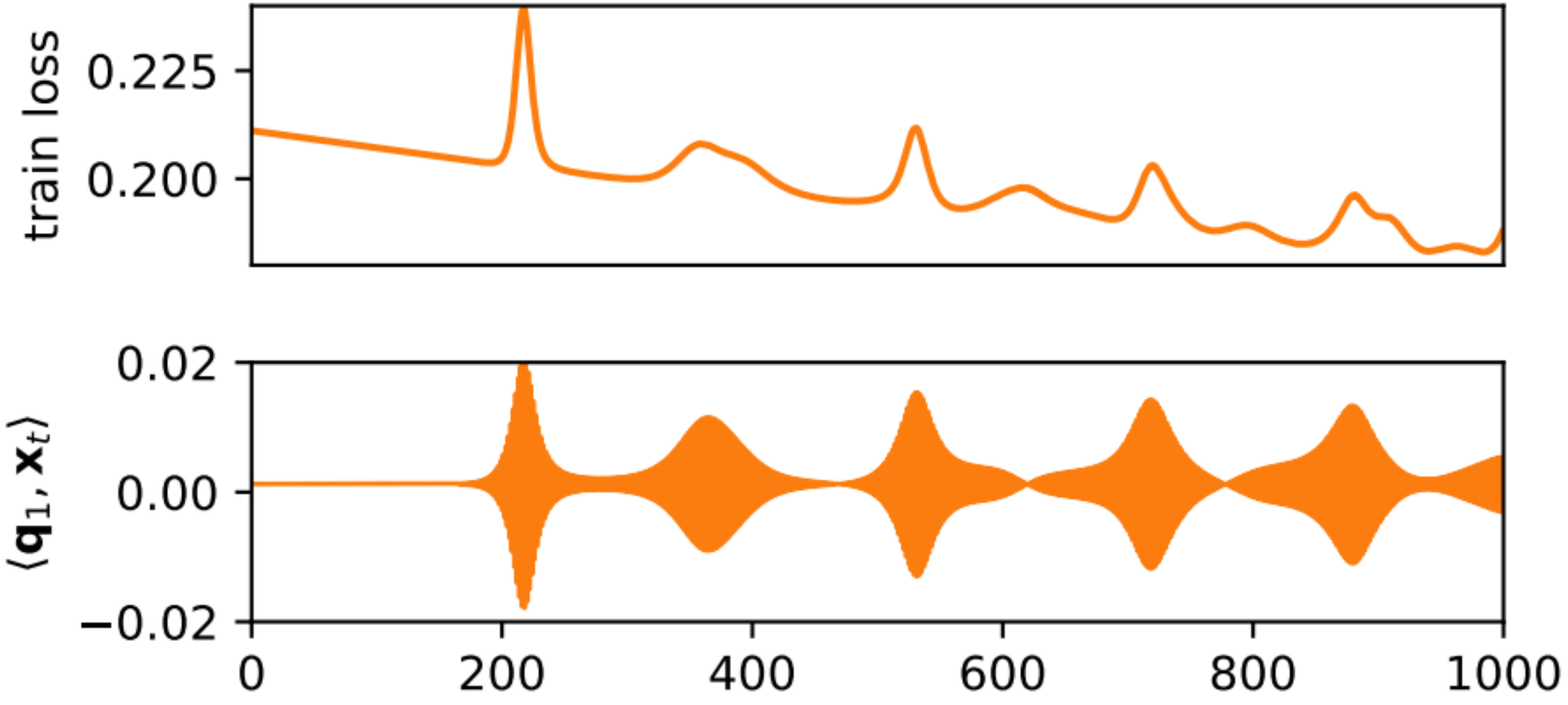
- Upon crossing the Edge of Stability, the loss begins to spike

- Correspondingly, weights iterate increasingly along $\mathbf{q}_1$



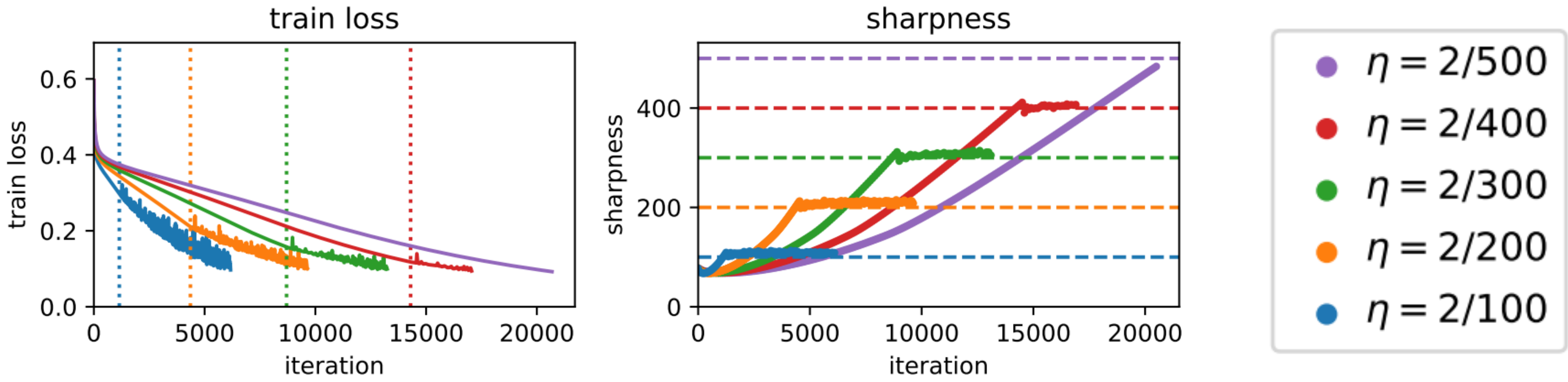
A Closer Look

- Later, things continue to descend again — non-monotonically



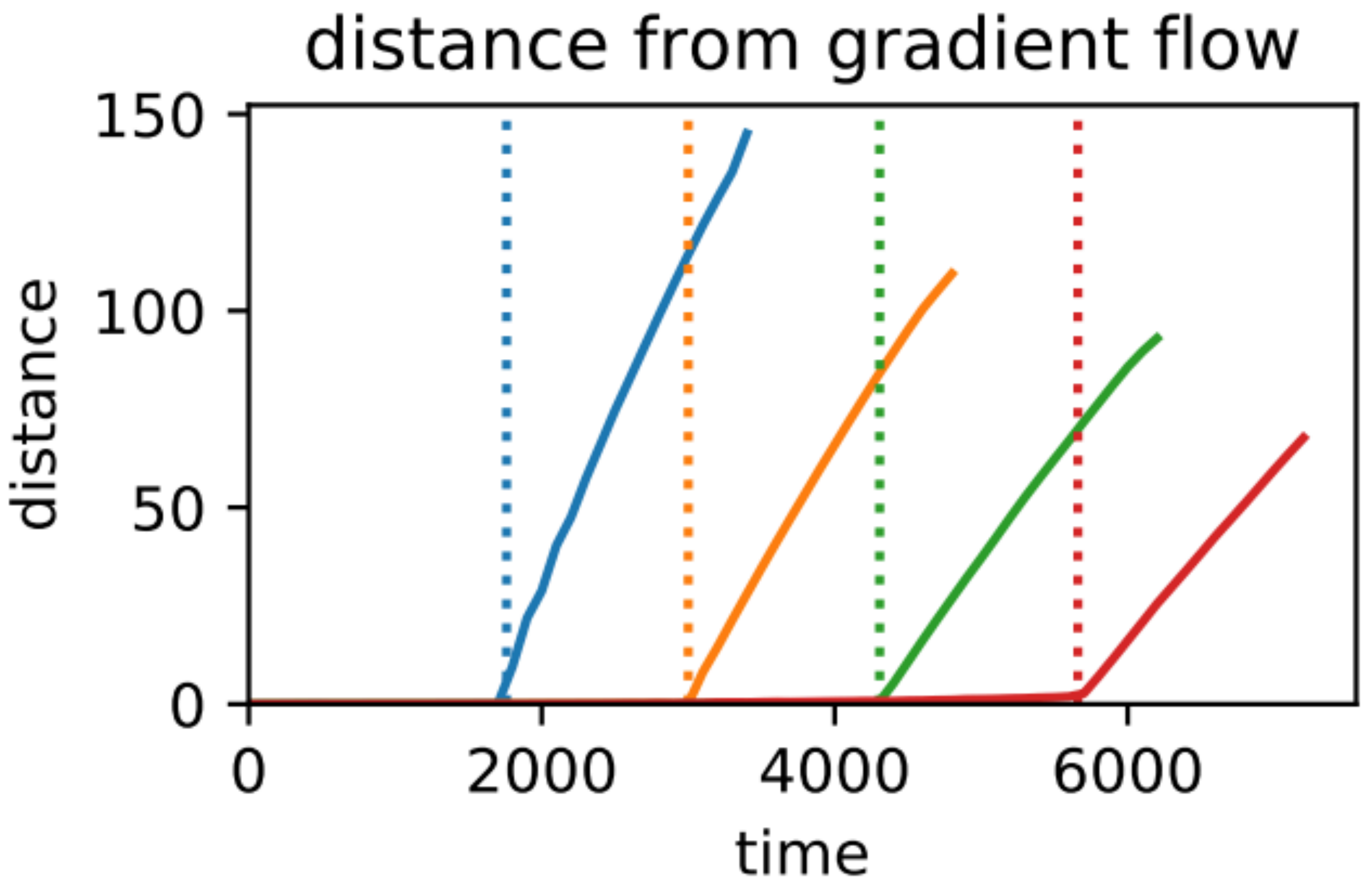
Stable η may be suboptimal

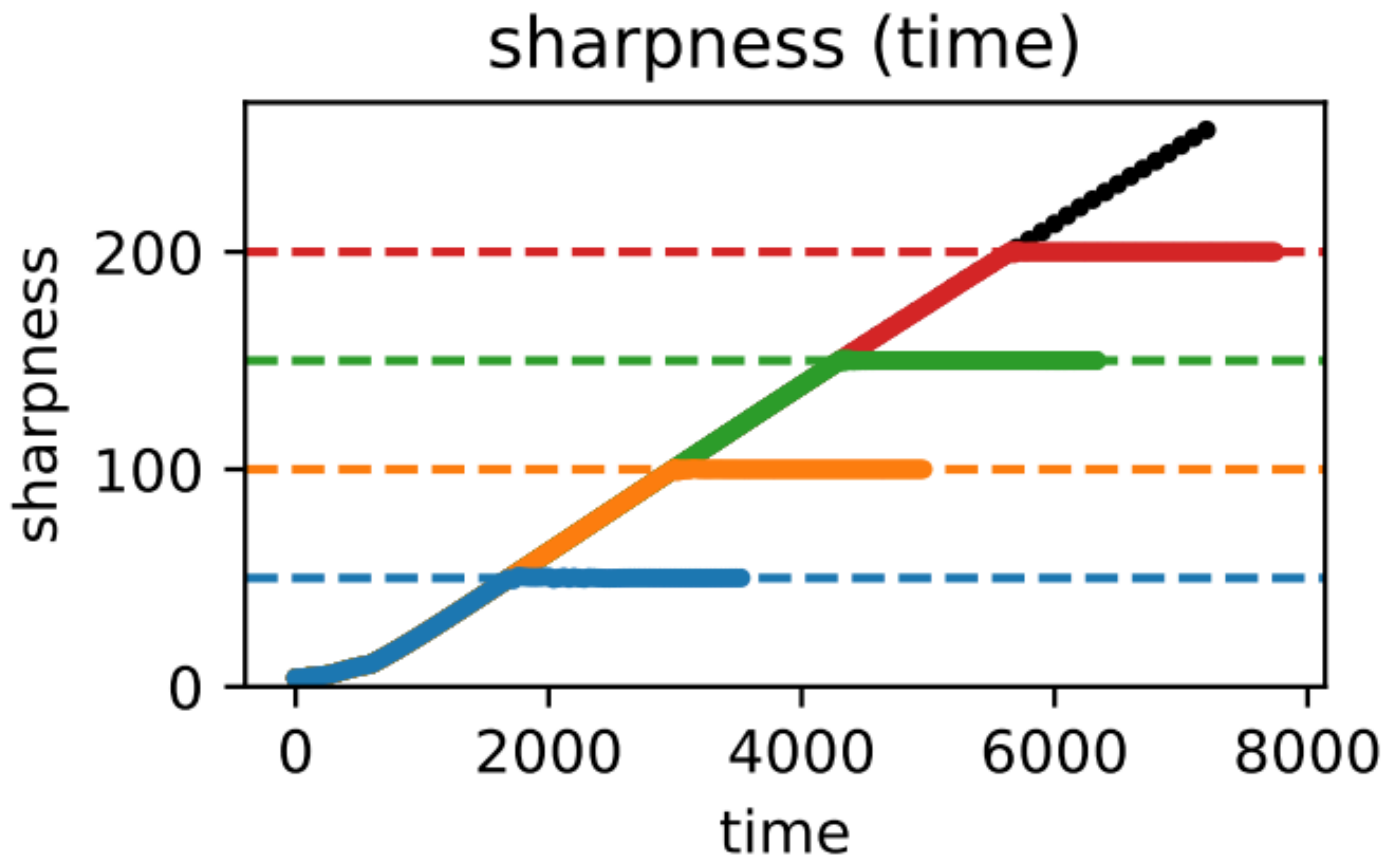
- Larger η trains faster despite prolonged Edge of Stability phase.
- Avoiding the Edge of Stability requires “unreasonably” small η



Gradient Flow

- Gradient Flow (GF) is when $\eta \to 0$
- When not in the Edge of Stability, GD tracks GF (but **not for all** networks)
- In these plots, $time = \eta \times iteration$





Prior Work, and relation to SGD

- Wu et al. 2018 “How sgd selects the global minima in over-parameterized learning: A dynamical stability perspective”
  - Sharpness at end of training is $\approx 2/\eta$
- Jastrzebski et al. 2020 “The break-even point on optimization trajectories of deep neural networks.”
  - Smaller step size, or larger batch size, leads SGD to end up in regions of larger sharpness
  - GD is a special case of SGD, but for SGD there may not be such a “tight” $2/\eta$ bound that the sharpness follows

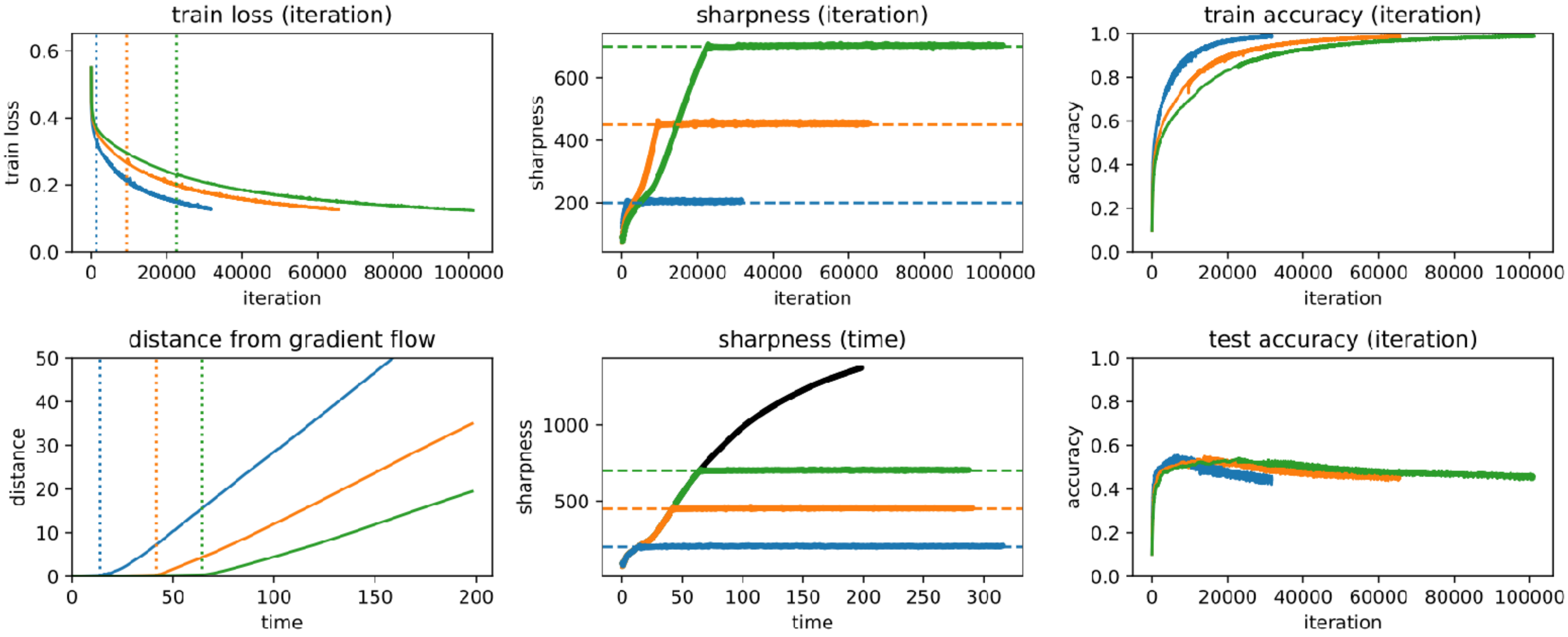
Basic Experimental Setup

- Fully-connected network, 2 hidden layers w/ width 200, tanh activations
- Full-batch gradient descent on 5K-subset of CIFAR-10
- Mean-Squared Error (MSE), and Cross-Entropy Loss
- This is the setup used for most figures in the main paper (shown in previous slides)

Architectural Choices

- 5K subset of CIFAR-10
- e.g Convolution with Max-pool

```
nn.Sequential(
    nn.Conv2d(3, 32, bias=True, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.MaxPool2d(2),
    nn.Conv2d(32, 32, bias=True, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.MaxPool2d(2),
    nn.Flatten(),
    nn.Linear(2048, 10, bias=True)
)
```

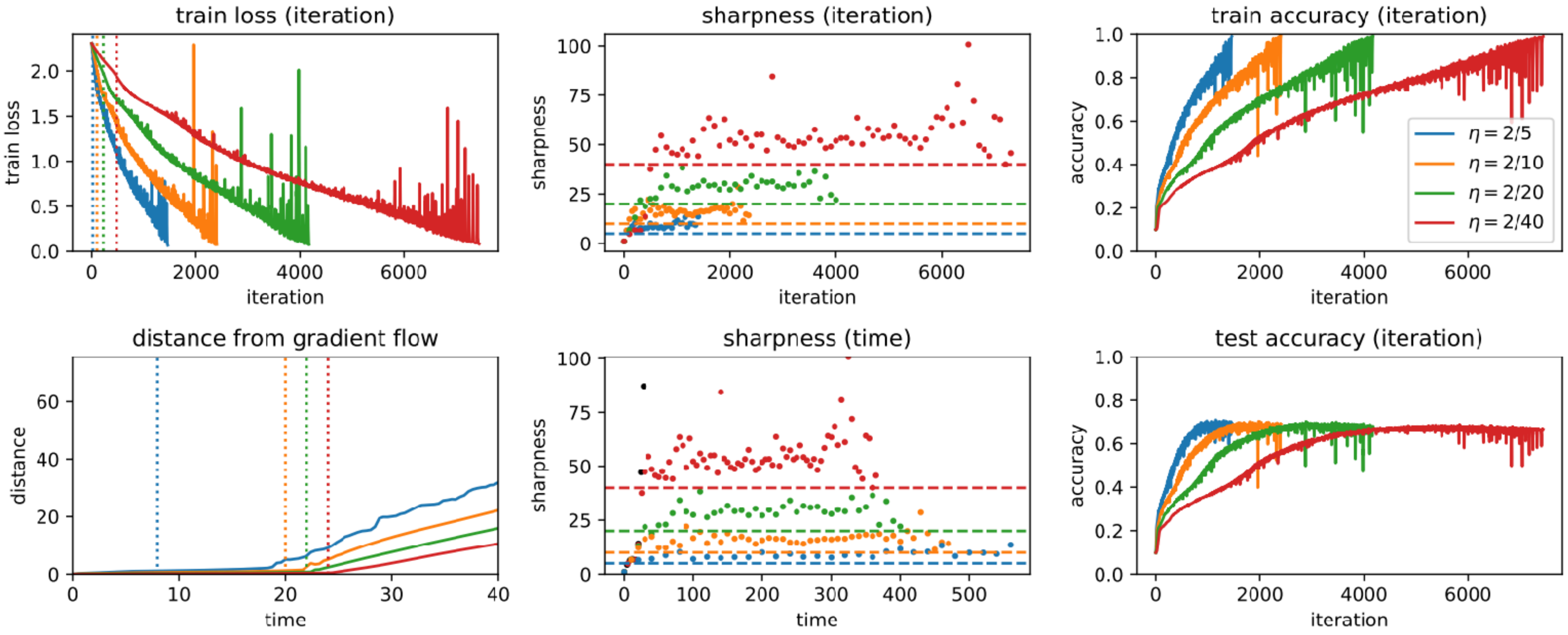


Architectural Choices

	Fully-Connected	Convolutional (Max-pool)	Convolutional (Average-pool)
tanh	✓	✓	✓
ReLU	✓	✓	✓
ELU	✓	✓	✓
softplus	✓	NA	NA
hardtanh	✓	NA	NA

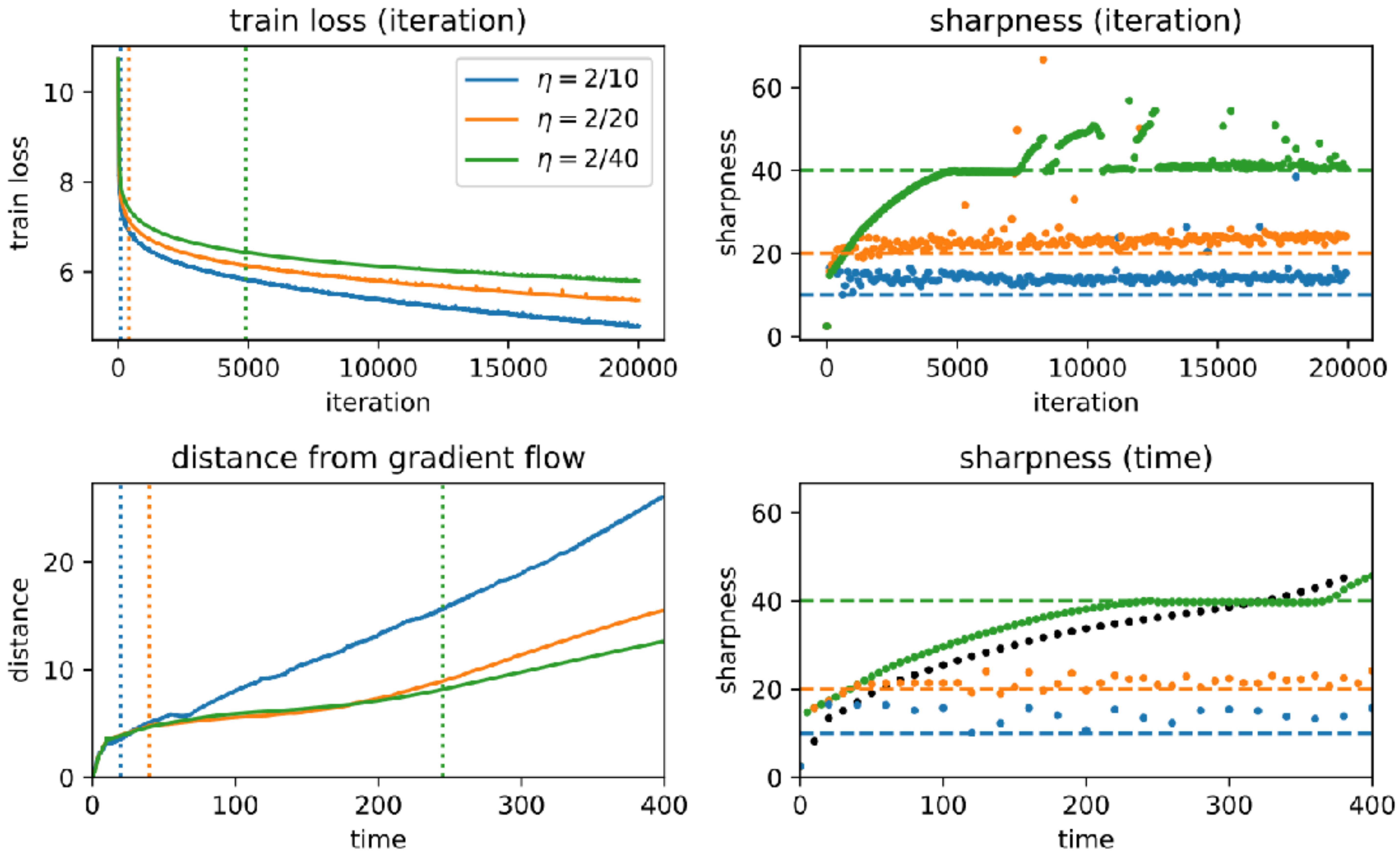
Standard Architectures

- Full CIFAR-10 dataset, with some approximation (ghost batching, statistics over 10% of dataset at a time)
- VGG-11 (with / without Batch Norm), ResNet-32 (shown below)



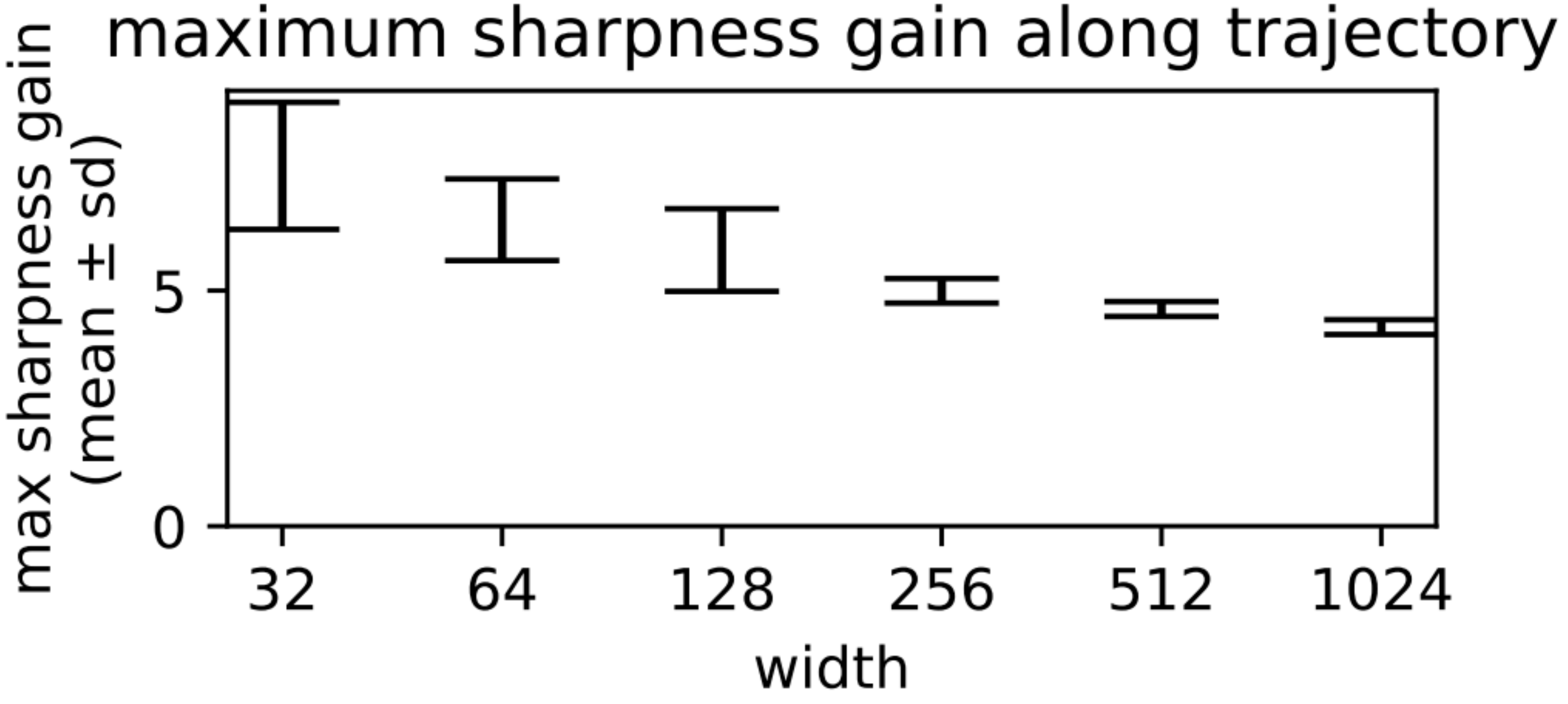
Additional Architectures: Transformer

- Transformer trained on WikiText-2 language modelling
- Gradient Flow is never tracked closely



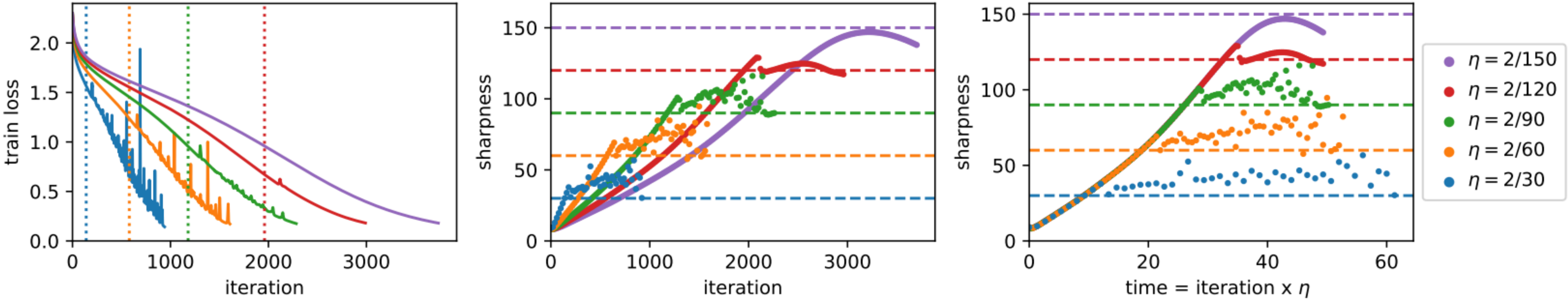
Network Width

- “*Wide networks of any depth evolve as linear models under gradient descent*” [Lee et al. 2020]
- $\mathbf{H}$ changes a vanishingly small amount as width increases
- “Sharpness gain” h_{max}/h_0 is lower as width increases (tends to 1?)



Cross-Entropy Loss

- Sharpness tends to lower near the end
- Authors investigate this by looking at the decomposition of $\mathbf{H}$

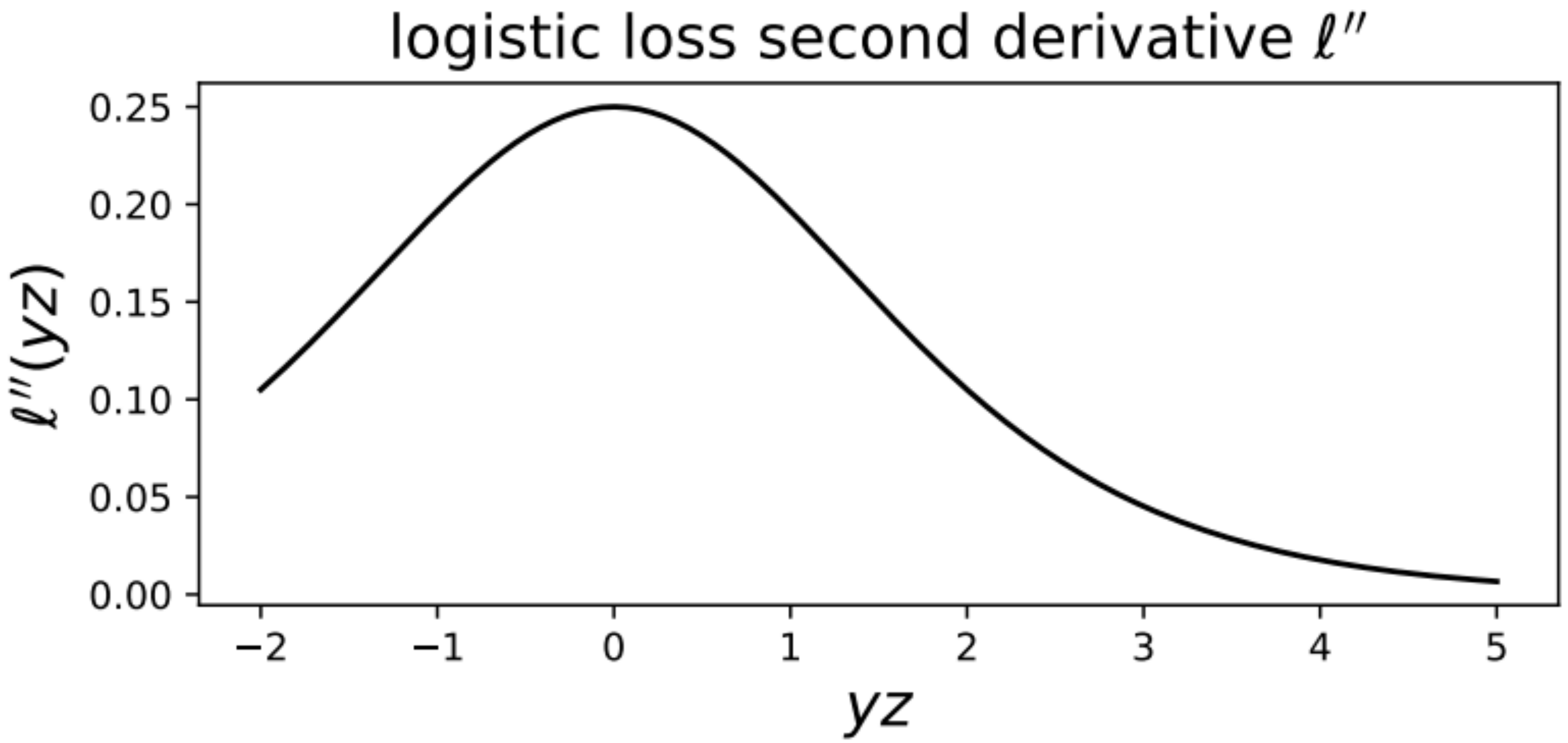


Cross-Entropy Loss

- Gauss-Newton approximation: $\mathbf{H} \approx \mathbf{J}_{\mathbf{zx}}^T \mathbf{H}_{\mathbf{z}} \mathbf{J}_{\mathbf{zx}}$ (verified empirically)
- Authors suggest that the *classical* Gauss-Newton $\mathbf{G} = \mathbf{J}_{\mathbf{zx}}^T \mathbf{J}_{\mathbf{zx}}$ still undergoes progressive sharpening
- For MSE loss $\mathbf{H}_{\mathbf{z}} = \mathbf{I}$, so $\mathbf{H}$ will experience the same sharpening trends as $\mathbf{G}$
- So maybe for Cross-Entropy, the flattening near the end is caused by a corresponding flattening of $\mathbf{H}_{\mathbf{z}}$

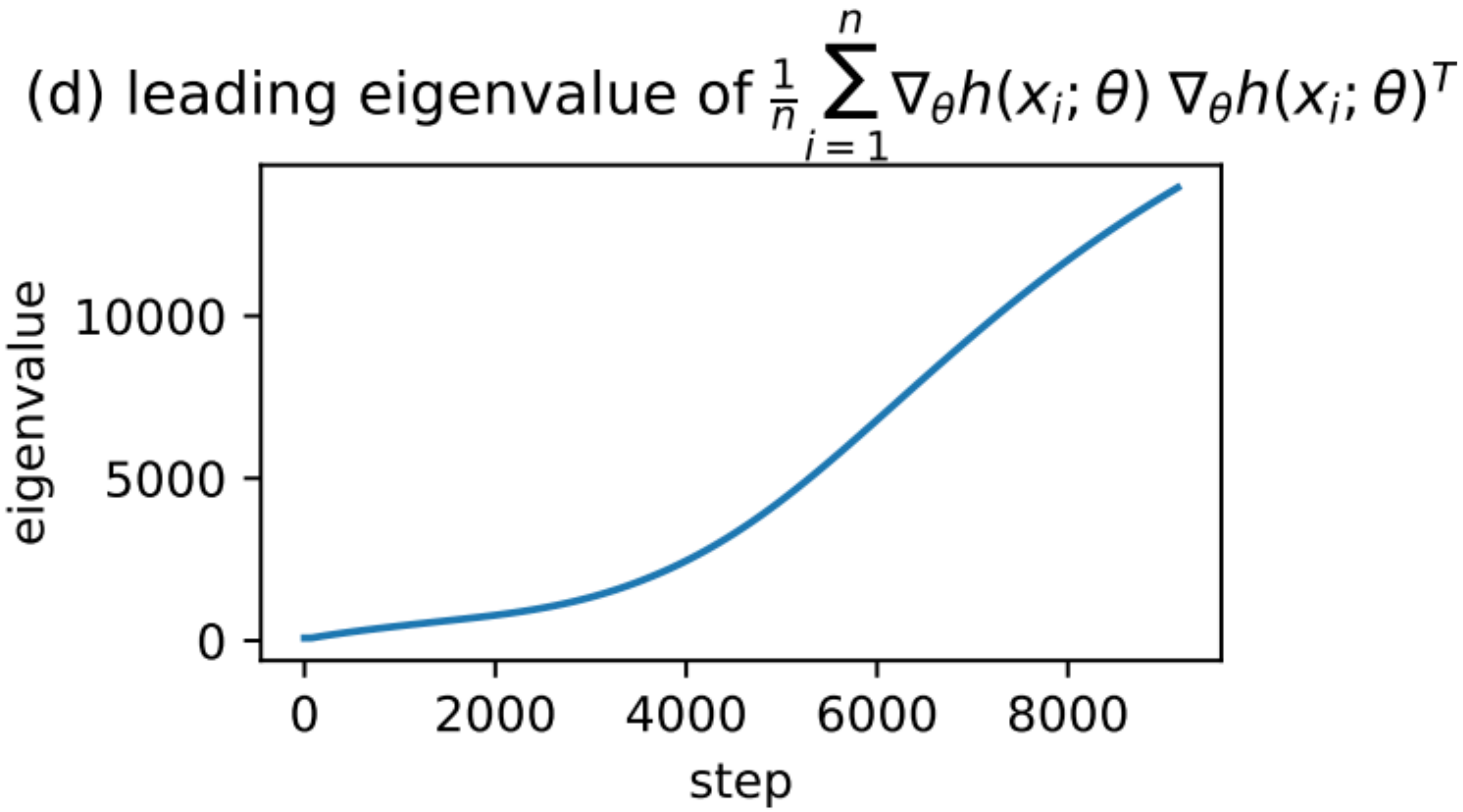
Cross-Entropy Loss

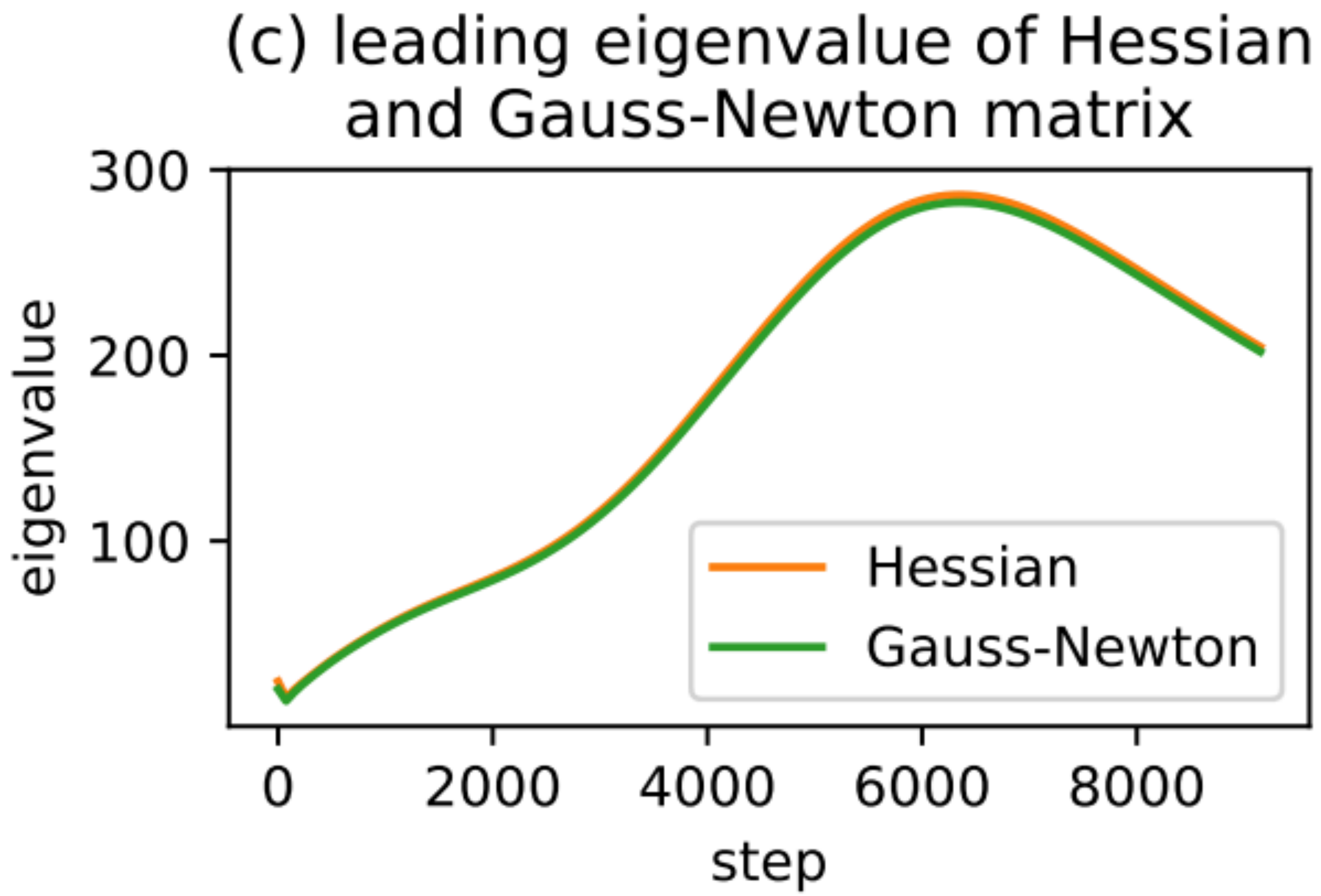
- Suppose we are using binary labels $y \in \{-1,1\}$
- As the model trains, logits z adopt same sign as y, and yz becomes larger
- Accordingly, the second derivative l'' w.r.t z decreases



Cross-Entropy Loss

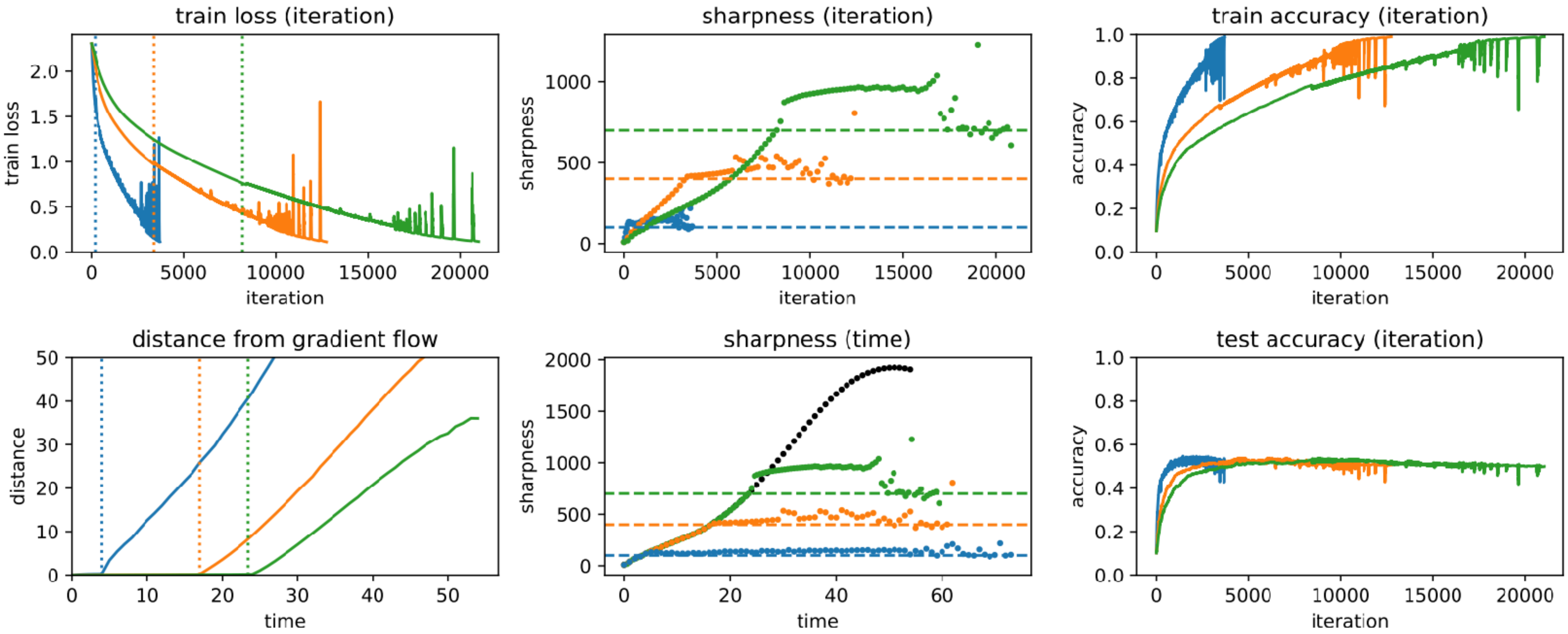
- Sharpness of classical Gauss-Newton $\mathbf{G}$ grows throughout training
- Attenuation of second derivative explains discrepancy
- The analysis is similar for multi-class case





Cross-Entropy Loss

- Compared to MSE, sharpness seems more random (why?)

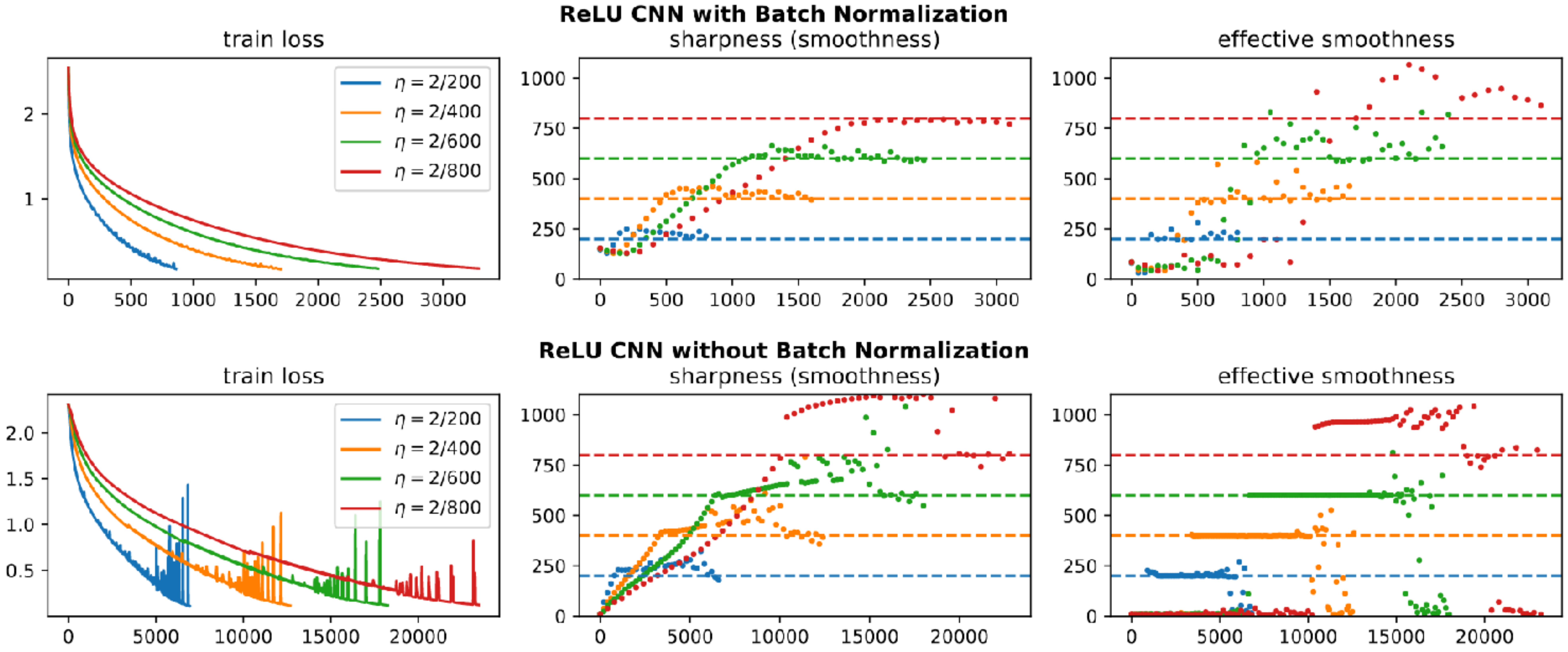


Batch Normalization

- “Effective smoothness”: maximum Lipschitz constant of gradient in update direction (scaled from 0 to α)
- In "*How does batch normalization help optimization?*” [Santurkar et al. 2018] argue that batch norm improves effective smoothness
- Authors argue against these results, saying that effectiveness smoothness may behave more regularly, but isn’t improved (there’s a difference in interpretation of the plots, shown on next slide).

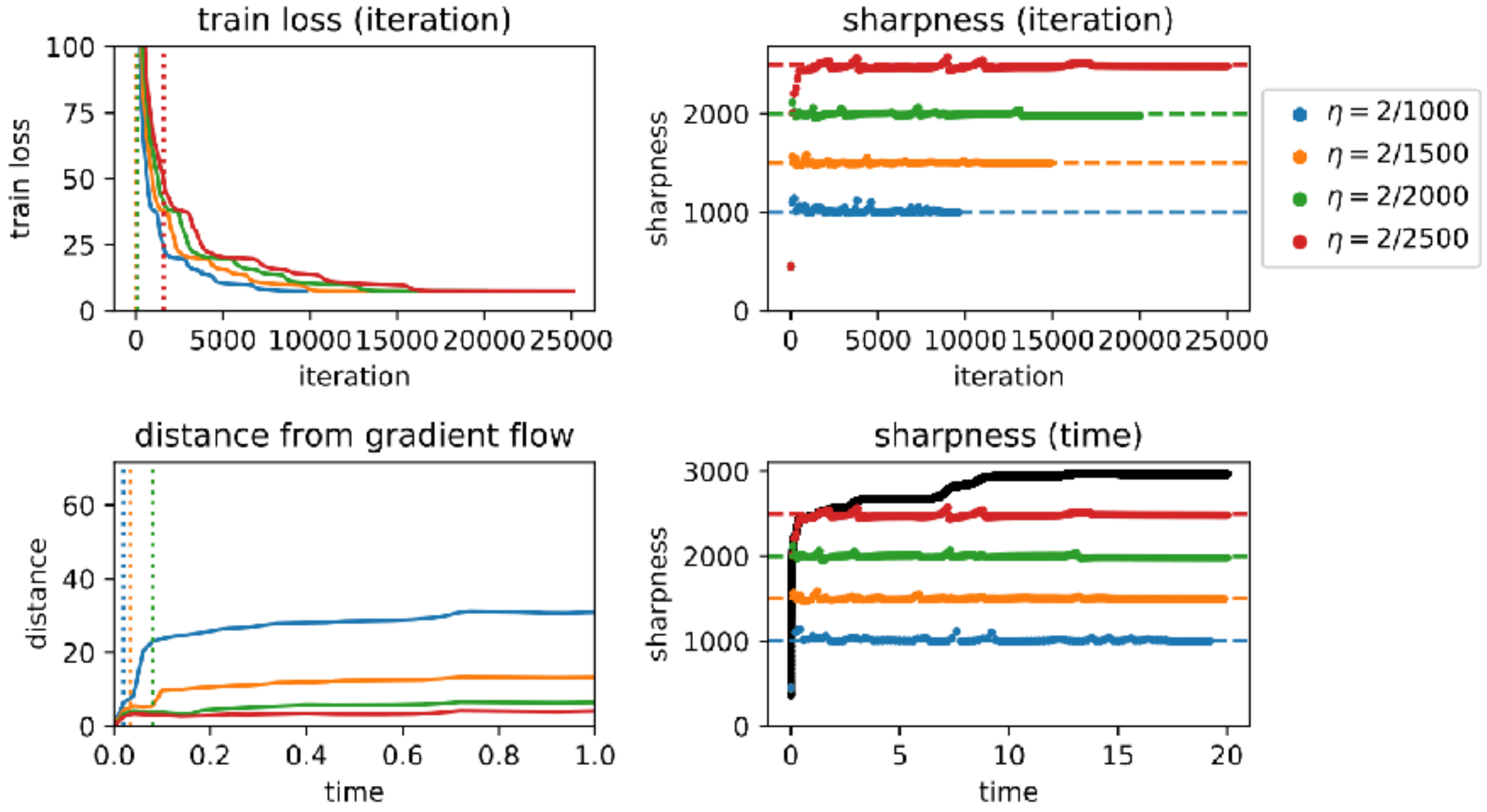
Batch Normalization

- *"… there is no evidence that the use of batch normalization improves either the smoothness or the effective smoothness …" (p. 56)*
- Is this interpretation reasonable?



Additional Architectures: Deep Linear Network

- *Deep linear network:* $f(\mathbf{x}) = \mathbf{W}_{20} \ldots \mathbf{W}_2 \mathbf{W}_1 \mathbf{x}$
- *Objective:* $\dfrac{1}{N} \sum_{i=1}^{N} \|f(\mathbf{x}_i) - \mathbf{y}_i\|_2^2$
- How deep does it actually need to be to show progressive sharpening?



Discussion

- Gradient descent (GD) optimizes loss while restraining itself from going into regions above a step size dependent sharpness
- Convergence analyses of GD that assume bounds on sharpness (L -smoothness) don't apply to reasonable step sizes
- With practical step sizes, GD does not monotonically decrease the loss
- The Edge of Stability is "non-quadratic": GD would quickly diverge (Appendix D) if we run it on an quadratic approximation
- The authors make no claims about sharpness and generalization, although this has been explored in e.g "*On large-batch training for deep learning: Generalization gap and sharp minima*" [Keskar et al. 2017]