Alternating Gradient Flows: A Theory of Feature Learning in Two-layer Neural Networks

Daniel Kunin[†]
Stanford University

Giovanni Luca Marchetti[†]
KTH

Feng Chen
Stanford University

Dhruva Karkada
UC Berkeley

James B. Simon
UC Berkeley and Imbue

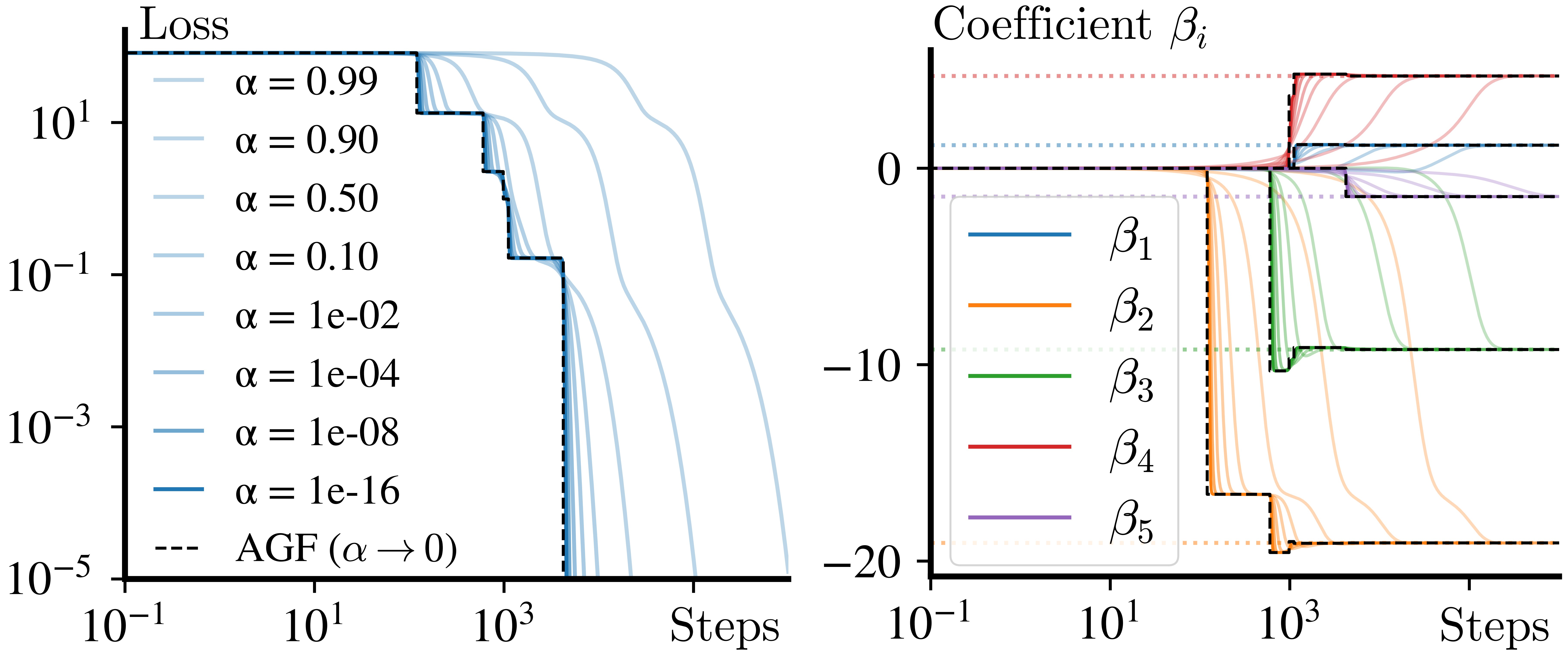
Michael R. DeWeese
UC Berkeley

Surya Ganguli
Stanford University

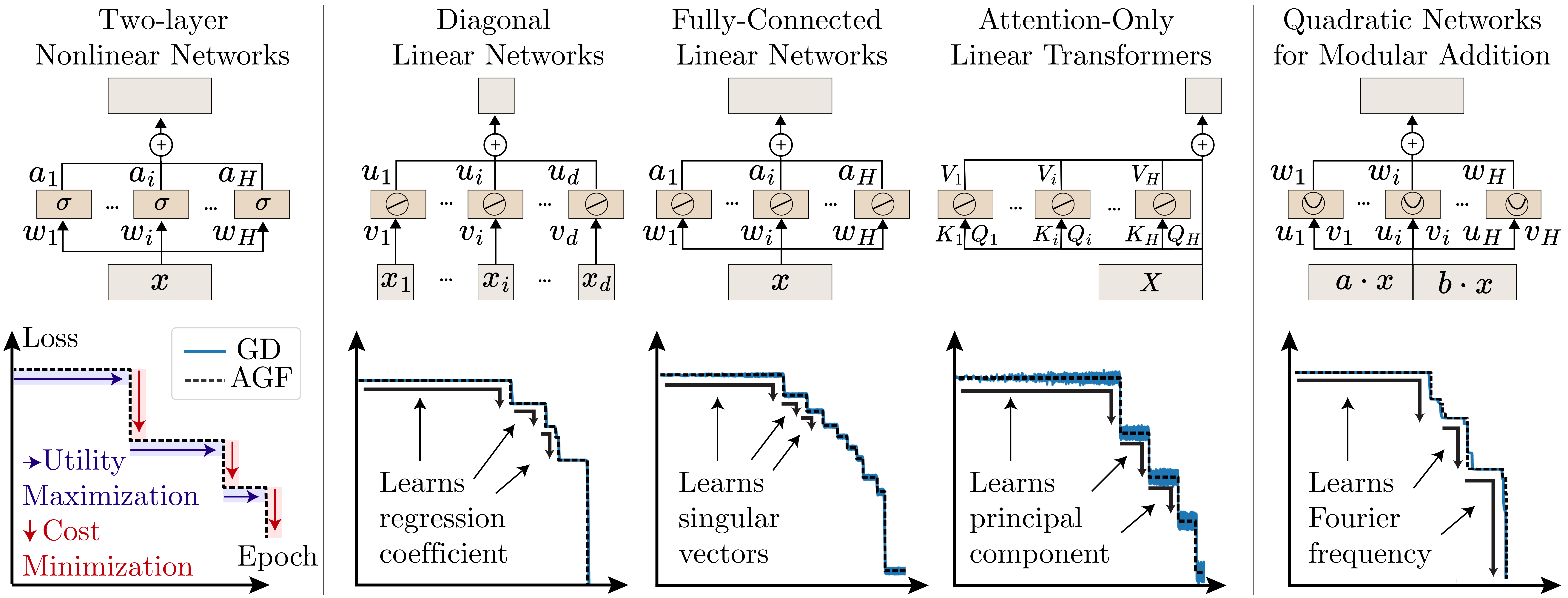
Nina Miolane
UC Santa Barbara

Kunin et al. (2025)

TL;DR: We introduce Alternating Gradient Flows, a framework modeling feature learning in <u>two-layer</u> networks with <u>vanishing initialization</u> as alternating utility maximization and cost minimization steps



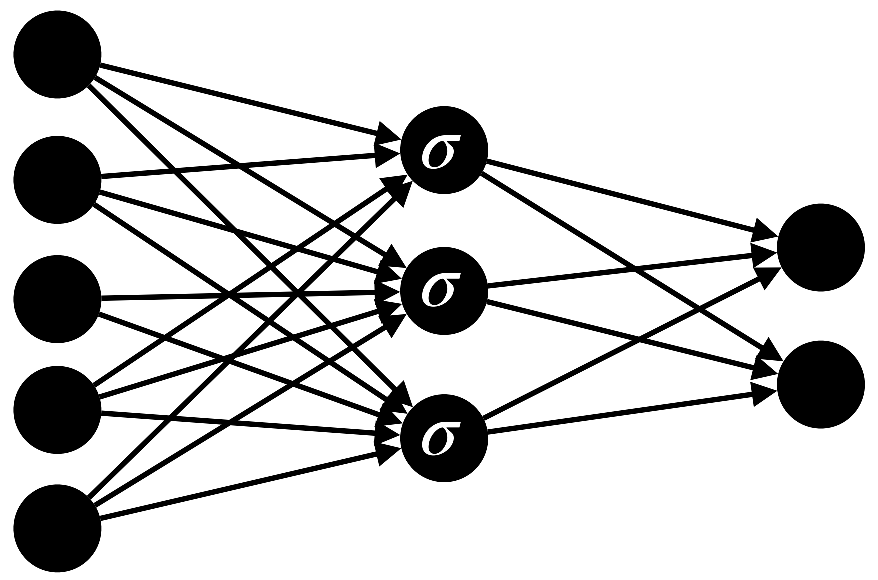
AGF — a step towards a broader understanding of feature learning



Part 1: Deriving a Two-step Algorithm (AGF) that Approximates Gradient Flow

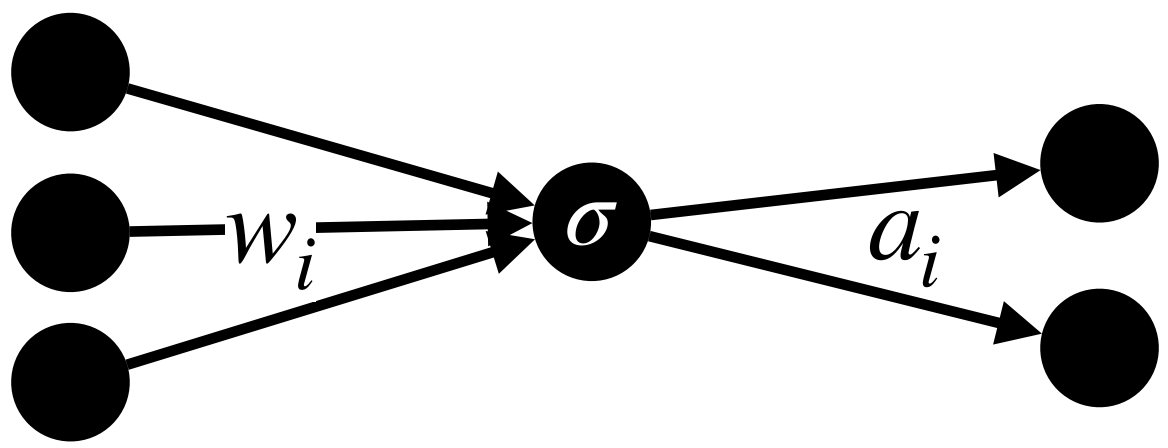
Two-layer network

$$f(x;\Theta) = \sum_{i=1}^{H} f_i(x;\theta_i)$$



"Neuron"

$$f_i(x;\theta_i) = a_i\sigma(w_i^\intercal x)$$

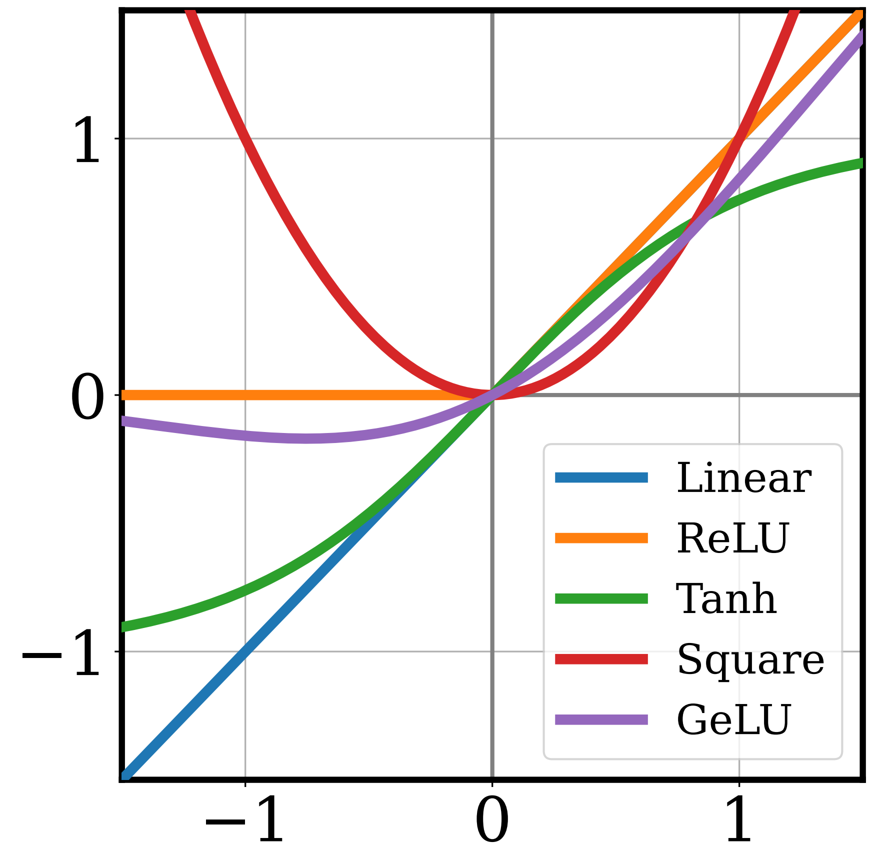


Parameters initialized

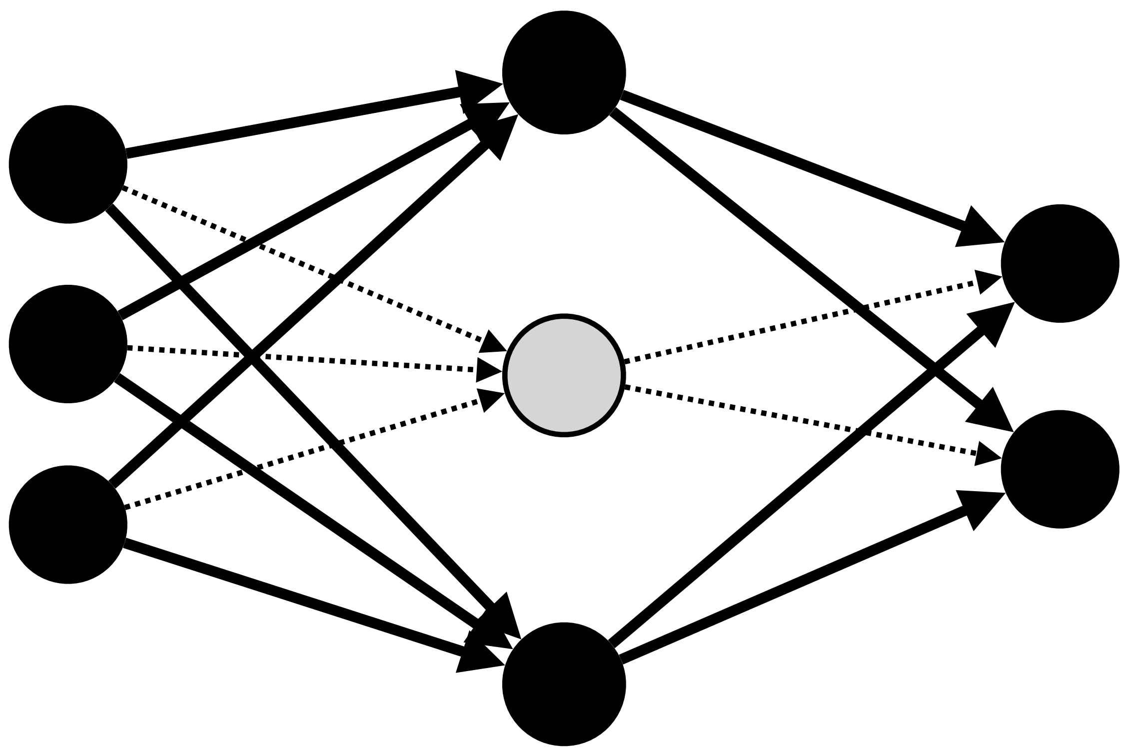
$\Theta_0 \sim \mathcal{N}(0,\alpha)$ with $\alpha \ll 1$

Origin-passing

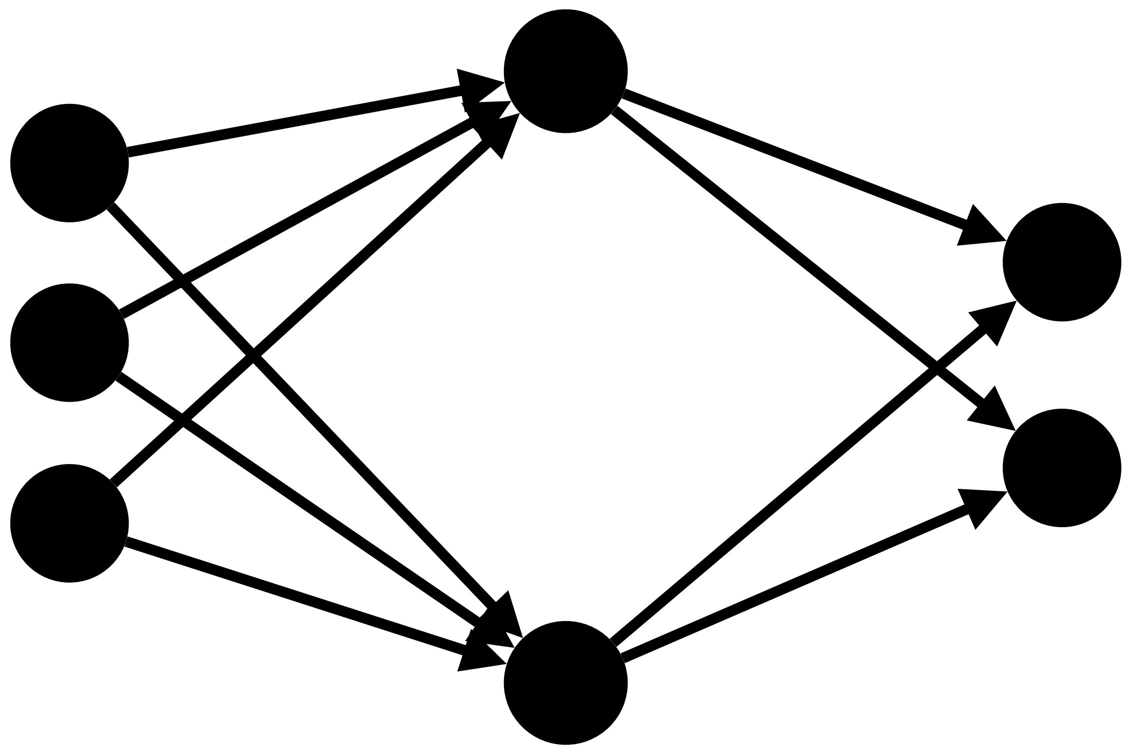
$$\sigma(0) = 0$$



Observation 1: If $\theta_i = 0$, then it will stay zero throughout training (i.e. $\nabla_{\theta_i}\mathscr{L}(\Theta) = 0$)



Observation 2: Neurons are only "aware" of other neurons through the residual $r(x;\Theta) = y(x) - f(x;\theta)$



Alternating Gradient Flows (AGF)

Initialize: $\mathscr{D} = [H], \mathscr{A} = \{\}, r(x) = y$

Utility Maximization:

For each $i \in \mathscr{D}$, Gradient ascent $\mathscr{U}_i(\theta_i, r)$ subject to $\|\theta_i\| = 1.$

Transition neuron with smallest jump time τ_i to active set.

Cost Minimization:

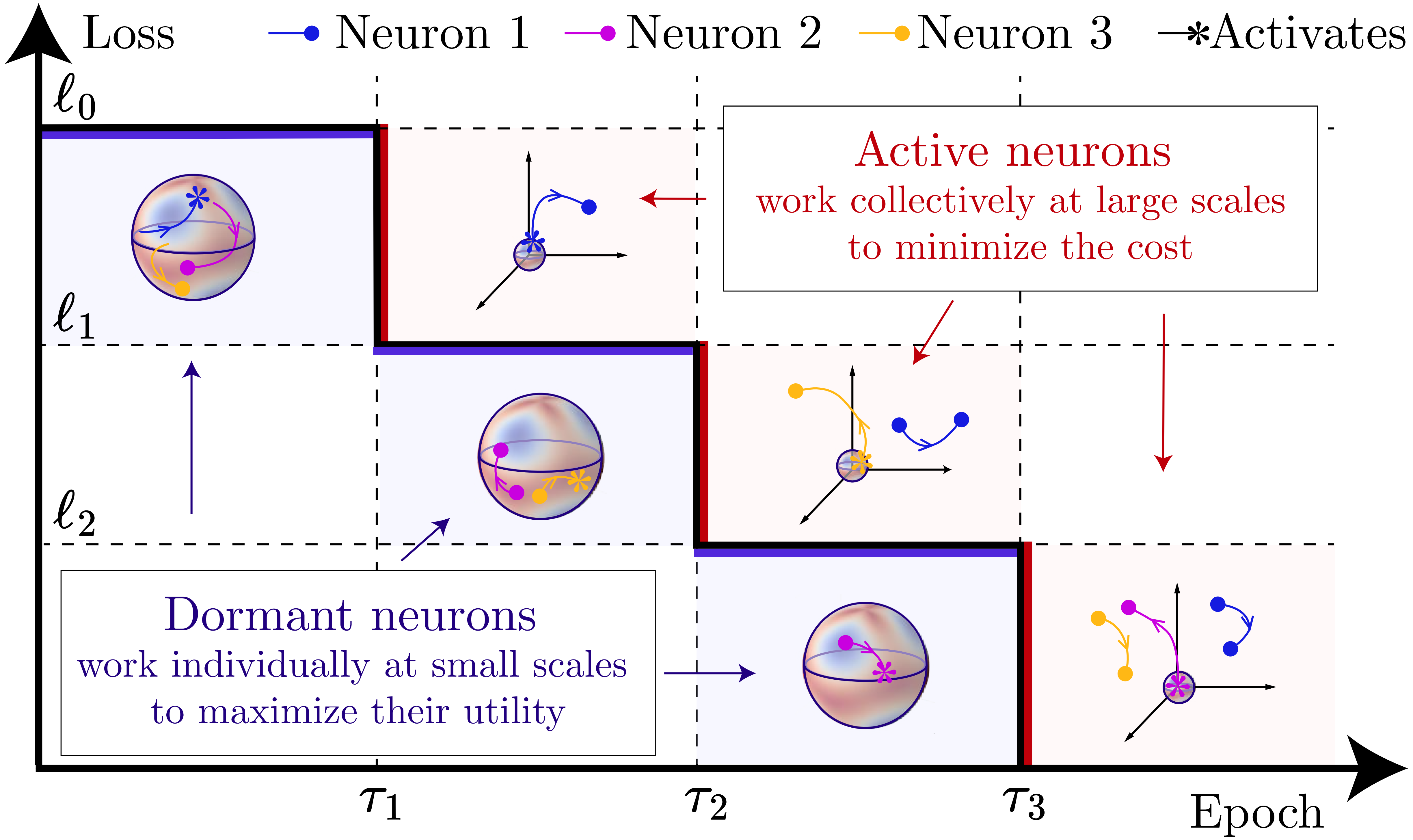
Gradient descent $\mathscr{L}(\Theta)$

subject to $\|\theta_i\| = 0, \quad \forall i \in \mathscr{D},$

$\|\theta_i\| \geq 0, \quad \forall i \in \mathscr{A}.$

Update residual and if necessary return collapsed neurons to dormant set.

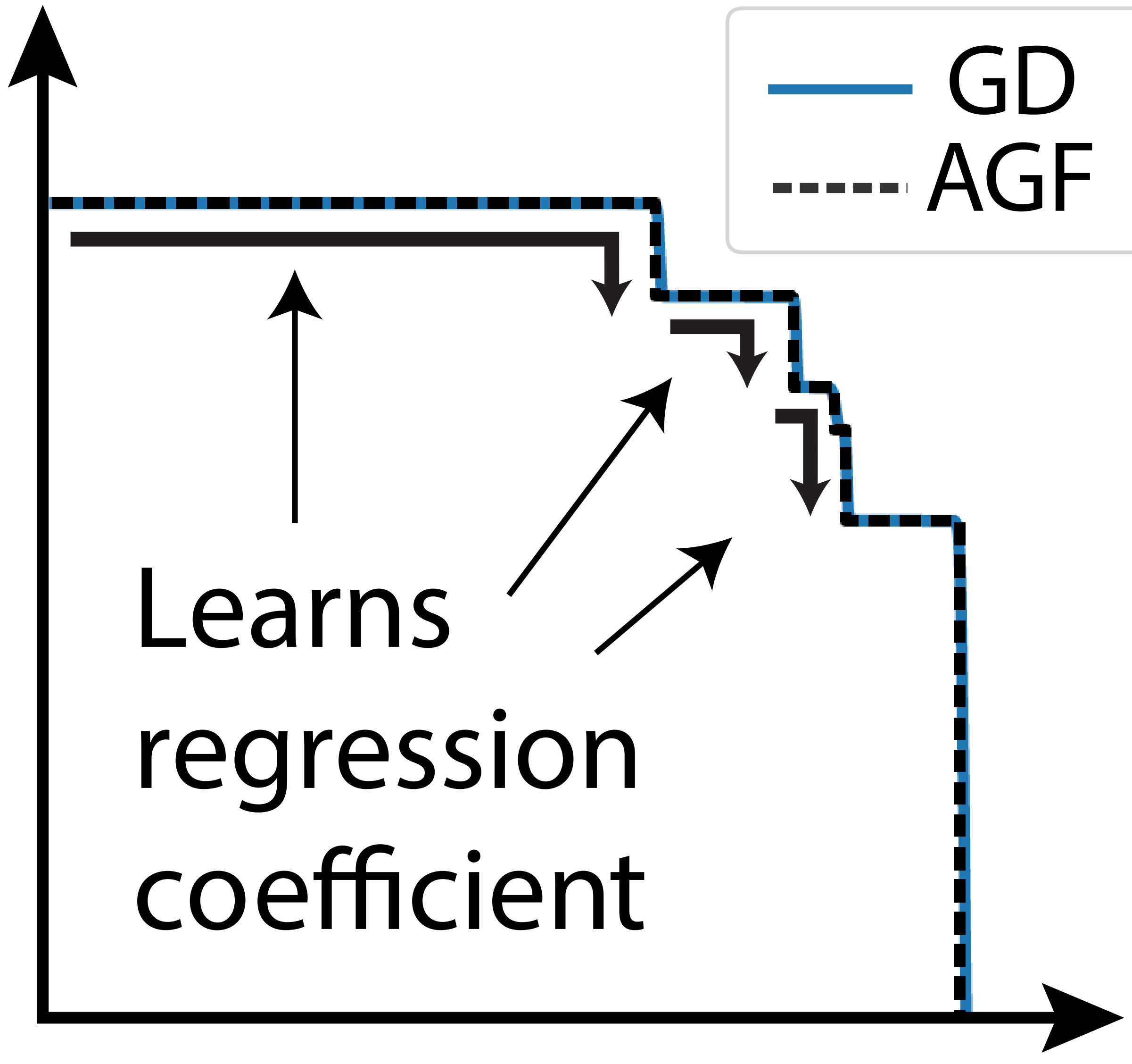
Termination: When $\mathscr{D} = \{\}$ or $r(x) = 0$



Part 2: AGF unifies many prior analysis of saddle-to-saddle dynamics

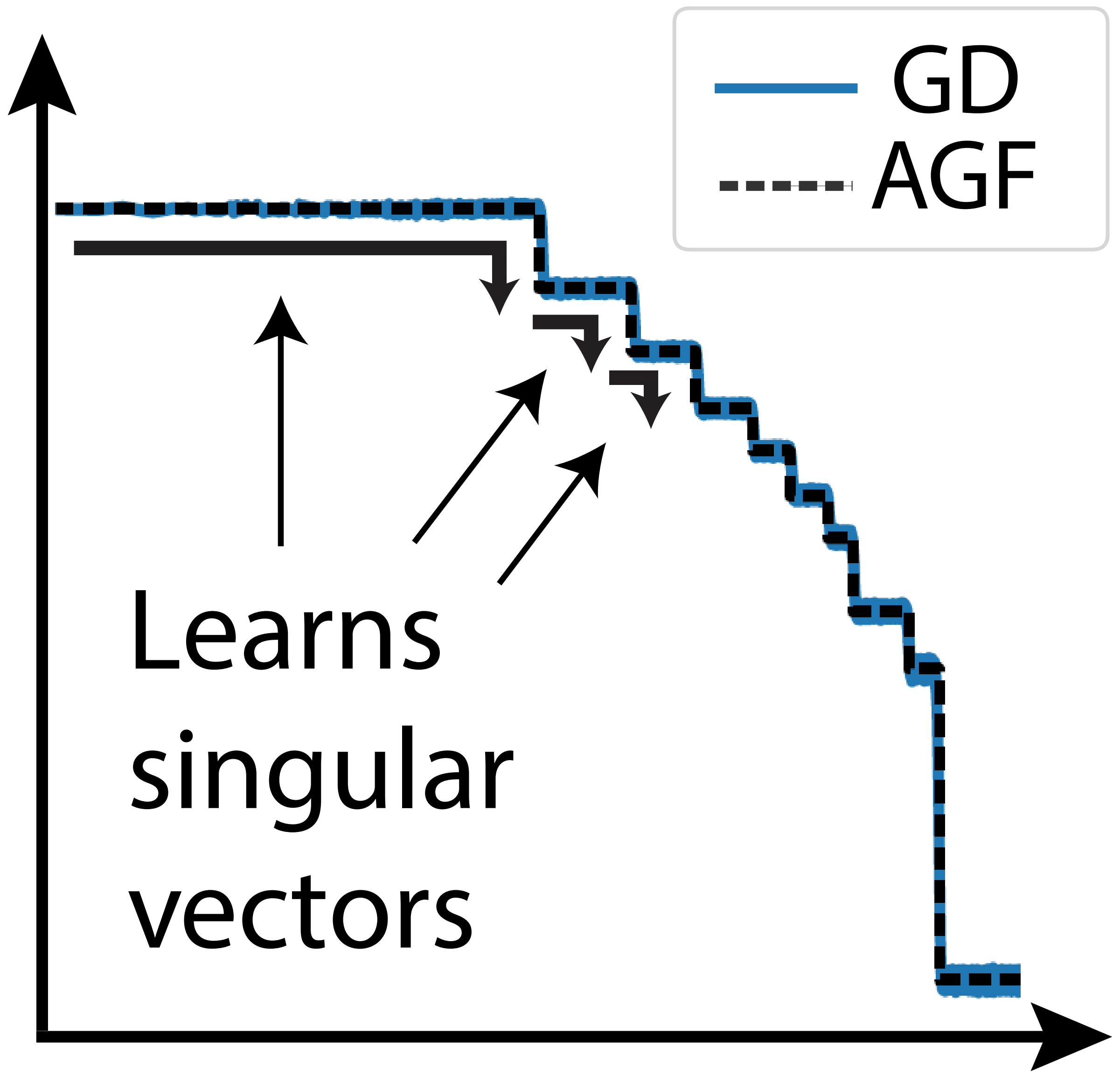
Diagonal Linear Networks

(Pesme & Flammarion, 2023)



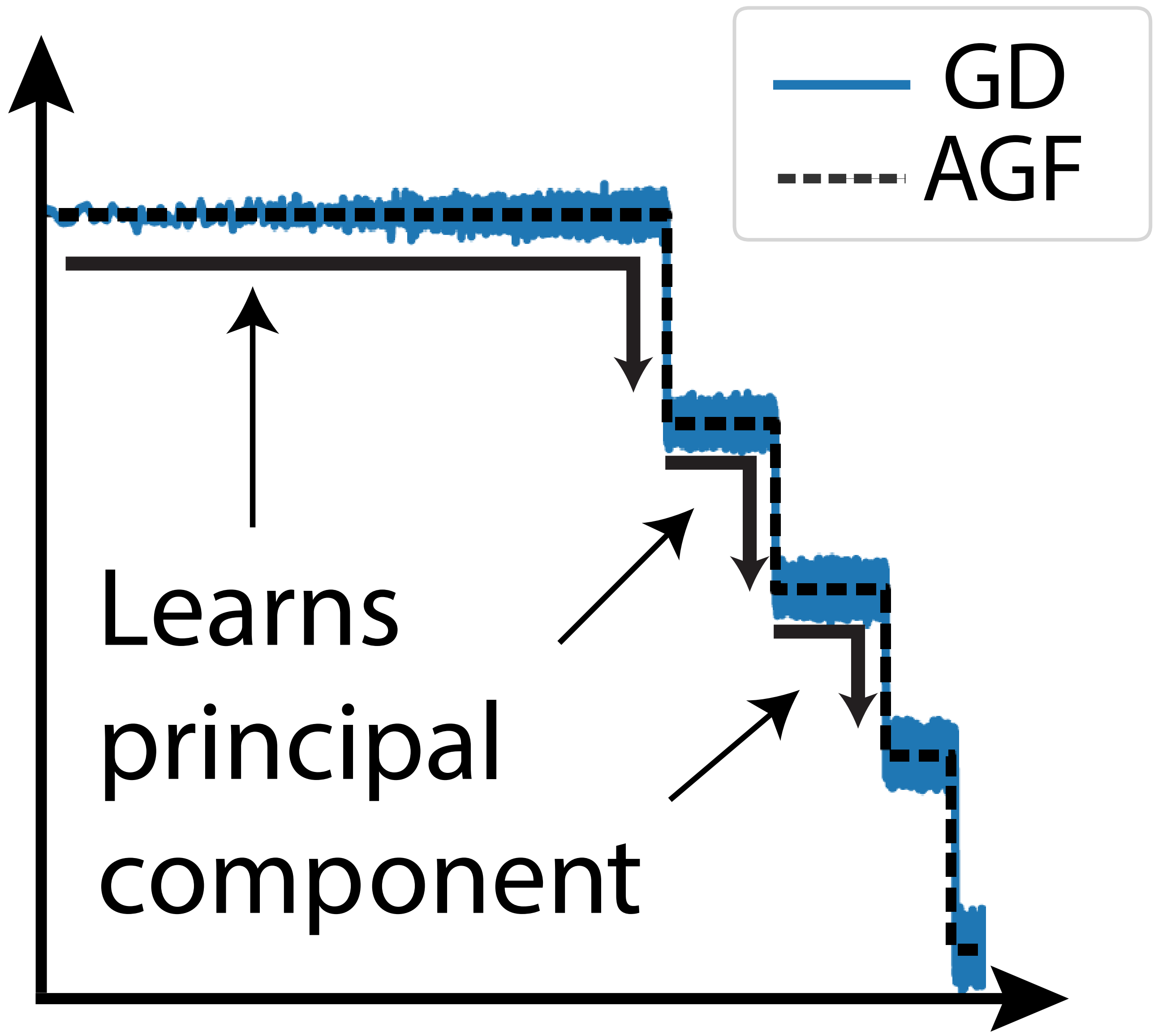
Fully Connected Linear Networks

(Saxe et al., 2014, Gidel et al., 2019, Li et al., 2020)



Attention-only Linear Transformers

(Zhang et al., 2025)



Part 3: AGF Explains the Emergence of Fourier Features in Modular Addition

PROGRESS MEASURES FOR GROKKING VIA MECHANISTIC INTERPRETABILITY

Neel Nanda[*, †] **Lawrence Chan**[‡] **Tom Lieberum**[†] **Jess Smith**[†] **Jacob Steinhardt**[‡]

Quantamagazine April 12, 2024

How Do Machines 'Grok' Data?

By apparently overtraining them, researchers have seen neural networks discover novel solutions to problems.

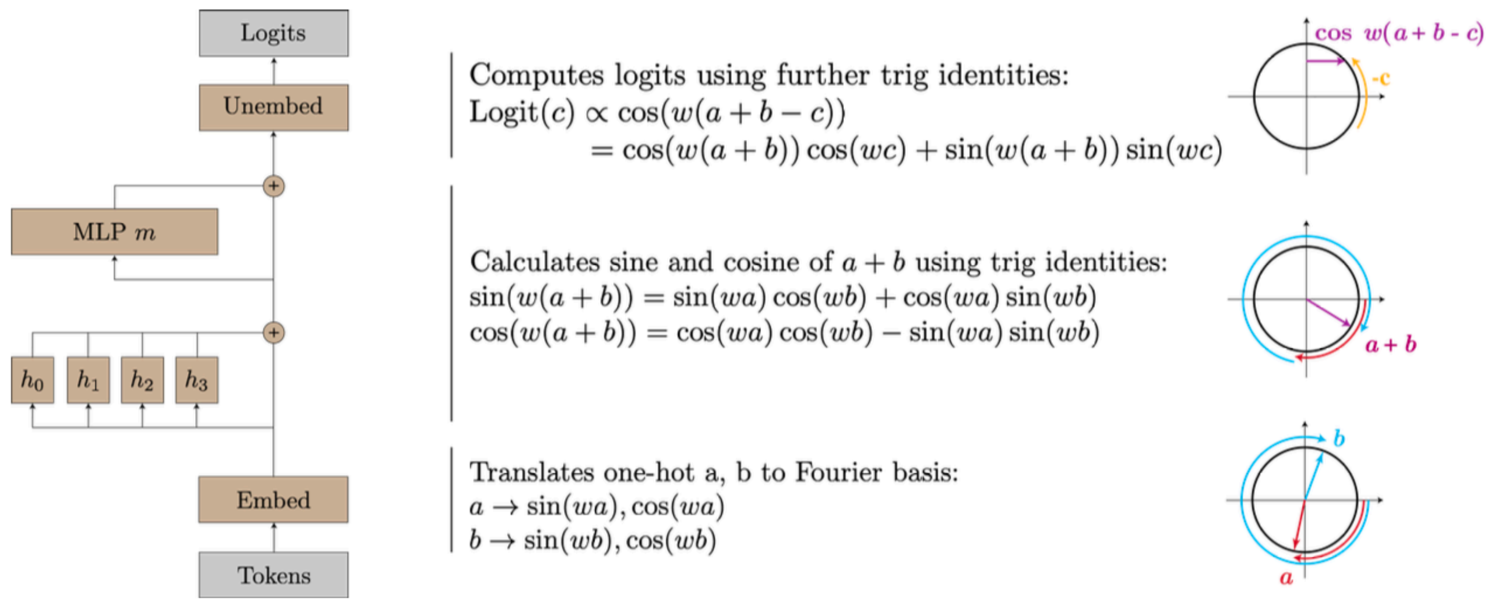
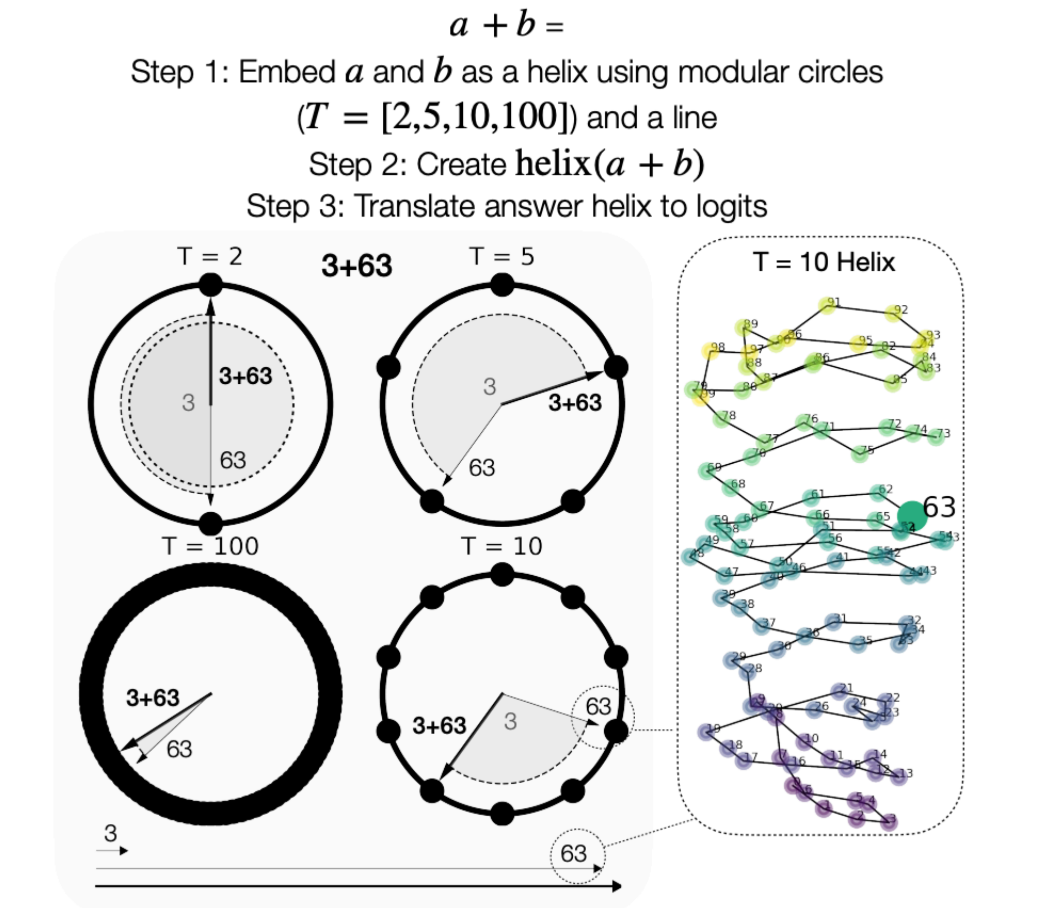


Figure 1: The algorithm implemented by the one-layer transformer for modular addition. Given two numbers a and b, the model projects each point onto a corresponding rotation using its embedding matrix. Using its attention and MLP layers, it then composes the rotations to get a representation of $a+b \mod P$. Finally, it "reads off" the logits for each $c \in \{0, 1, ..., P-1\}$, by rotating by $-c$ to get $\cos(w(a+b-c))$, which is maximized when $a+b \equiv c \mod P$ (since w is a multiple of $\frac{2\pi}{P}$).



Pre-trained Large Language Models Use Fourier Features to Compute Addition

Tianyi Zhou Deqing Fu Vatsal Sharan Robin Jia
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
{tzhou029,deqingfu,vsharan,robinjia}@usc.edu

Language Models Use Trigonometry to Do Addition

Subhash Kantamneni[1] **Max Tegmark**[1]

A striking example of feature learning, prior works have shown networks trained to perform **modular addition** learn internal **Fourier features** and use trigonometric identities to implement **addition as rotations** on the circle

1. Utility Maximization - neurons specialize to a dominant frequency

We prove that the utility-maximizing unit vectors are cosine waves at the dominant frequency of the encoding vector.

Theorem F.3. *Let ξ be a frequency that maximizes $|\hat{x}[k]|$, $k = 1, \dots, p-1$, and denote by s_x the phase of $\hat{x}[\xi]$. Then the unit vectors $\theta_* = (u_*, v_*, w_*)$ that maximize the utility function $\mathcal{U}(\theta; y)$ take the form*

$$\begin{aligned} u_*[a] &= \sqrt{\frac{2}{3p}} \cos\left(2\pi \frac{\xi}{p} a + s_u\right) \\ v_*[b] &= \sqrt{\frac{2}{3p}} \cos\left(2\pi \frac{\xi}{p} b + s_v\right) \\ w_*[c] &= \sqrt{\frac{2}{3p}} \cos\left(2\pi \frac{\xi}{p} c + s_w\right), \end{aligned} \tag{58}$$

where $a, b, c \in \{0, \dots, p-1\}$ are indices and $s_u, s_v, s_w \in \mathbb{R}$ are phase shifts satisfying $s_u + s_v \equiv s_w + s_x \pmod{2\pi}$. They achieve a maximal value of $\bar{\mathcal{U}}^* = \sqrt{2/(27p^3)}|\hat{x}[\xi]|^3$. Moreover, the utility function has no local maxima other than the ones described above.

2. Cost Minimization - group of neurons collaborate to minimize loss

We prove that frequency aligned neurons remain aligned and that a group of aligned neurons must distribute their phase shifts to remove that frequency from the residual

Theorem F.6. *We have the following lower bound:*

$$\mathcal{C}(\Theta) - \mathcal{U}_0(\Theta) \geq -\frac{|\hat{x}[\xi]|^2}{p}. \tag{78}$$

Moreover, equality holds if, and only if, we have that $\sum_{i=1}^N C_i \cos(\alpha_i) = \sum_{i=1}^N C_i \sin(\alpha_i) = 0$ for any choice of (C_i, α_i) among

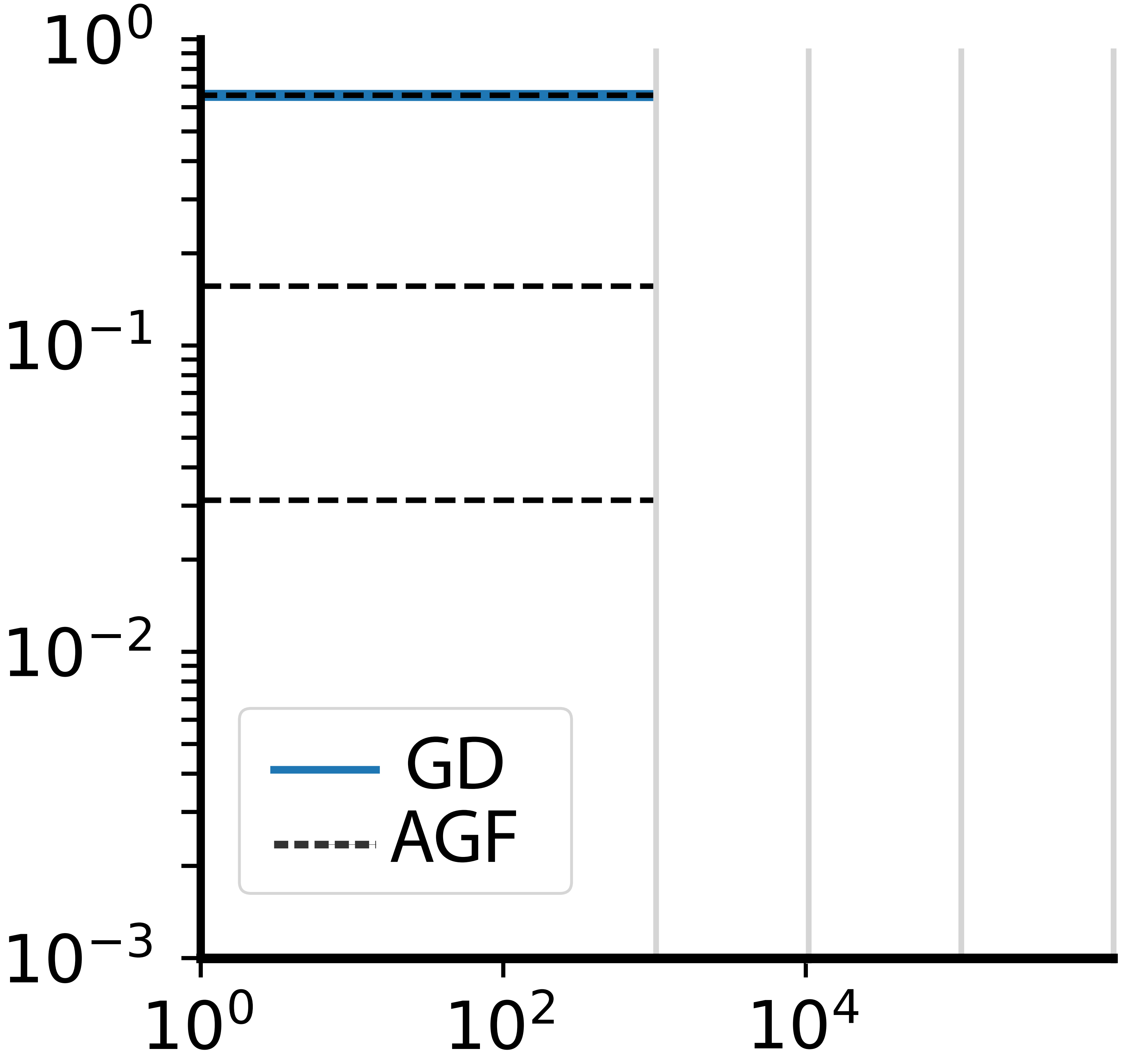
$$\begin{aligned} &(A_w^i((A_u^i)^2 + (A_v^i)^2), s_w^i), \\ &(A_w^i(A_u^i)^2, s_w^i \pm 2s_u^i), \\ &(A_w^i(A_v^i)^2, s_w^i \pm 2s_v^i), \\ &(A_w^i A_u^i A_v^i, s_w^i \pm (s_u^i - s_v^i)), \\ &(A_w^i A_u^i A_v^i, s_w^i + s_u^i + s_v^i), \end{aligned} \tag{79}$$

and, moreover, $\sum_{i=1}^N A_w^i A_u^i A_v^i \sin(s_w^i + s_x - s_u^i - s_v^i) = 0$ and $\sum_{i=1}^N A_w^i A_u^i A_v^i \cos(s_w^i + s_x - s_u^i - s_v^i) = \sqrt{54p}/|\hat{x}[\xi]|$.

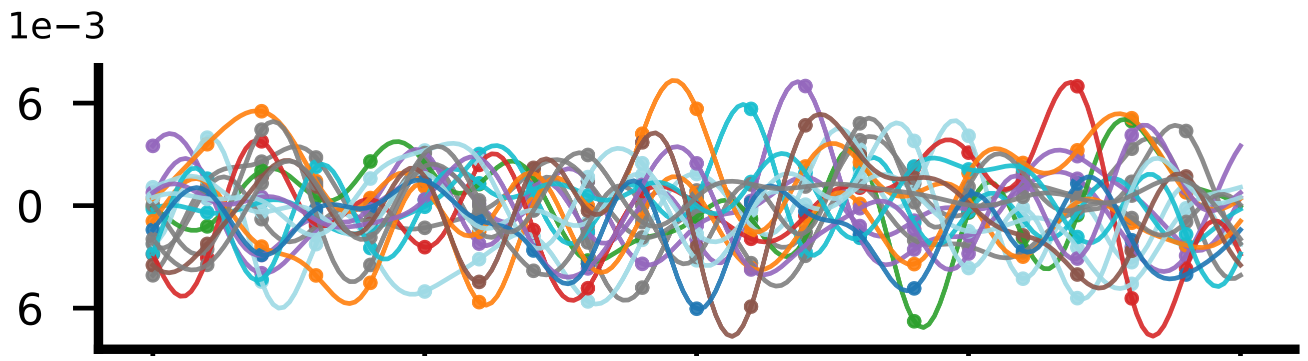
Putting it together: Greedy Fourier decomposition

Parameter Space Explanation

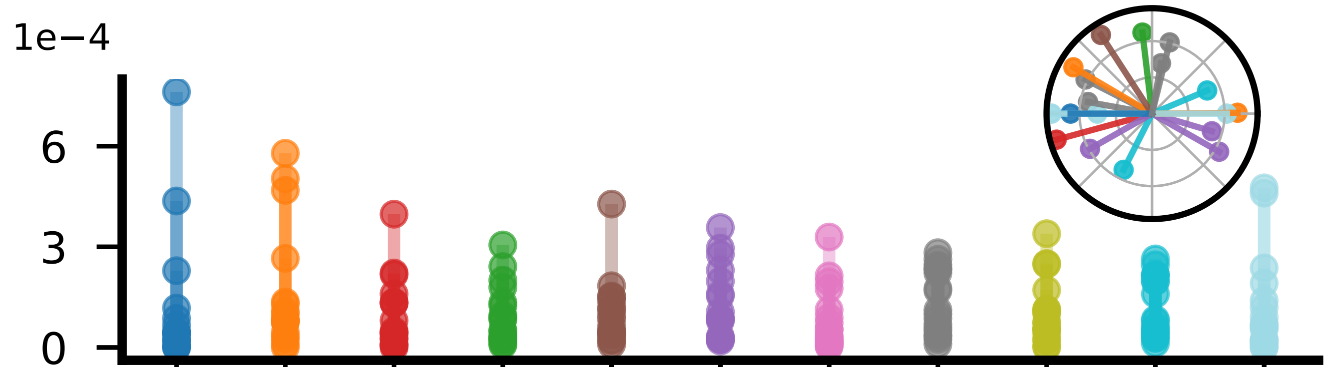
Train Loss



Weights w_i

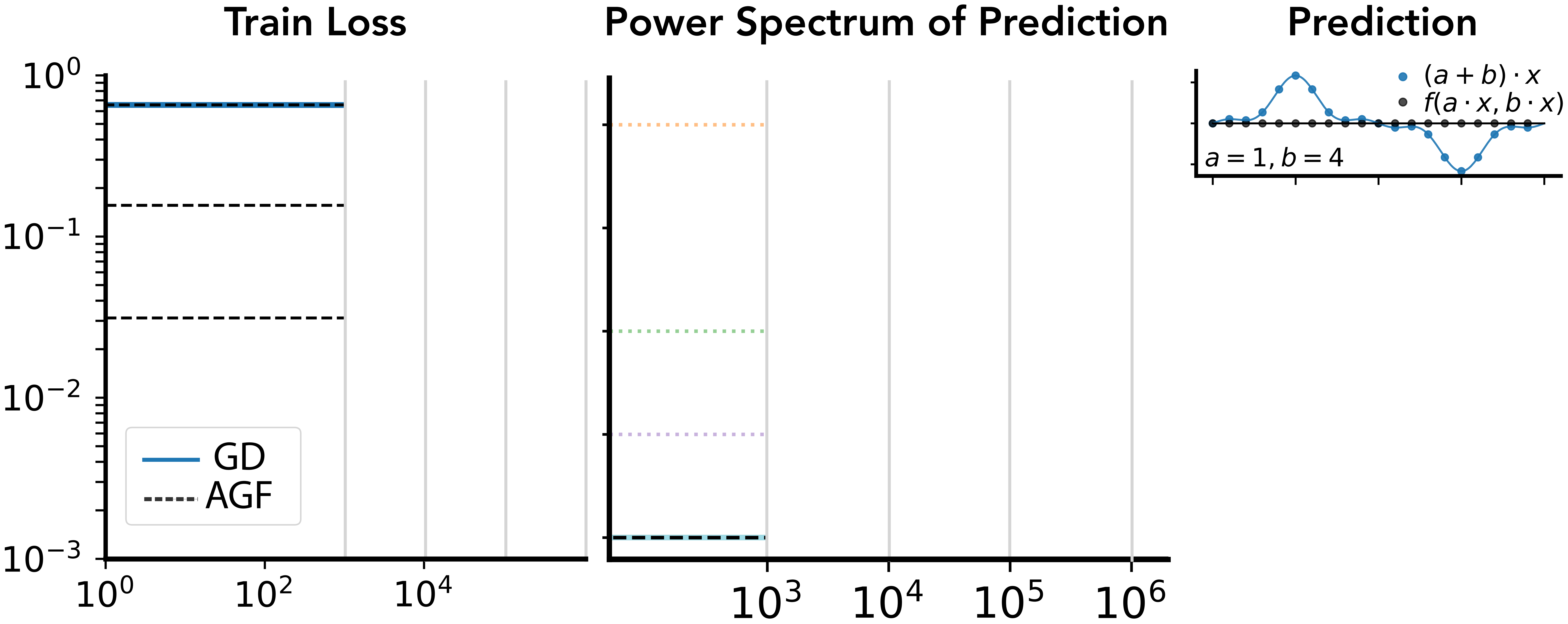


Frequency and Phase



Putting it together: Greedy Fourier decomposition

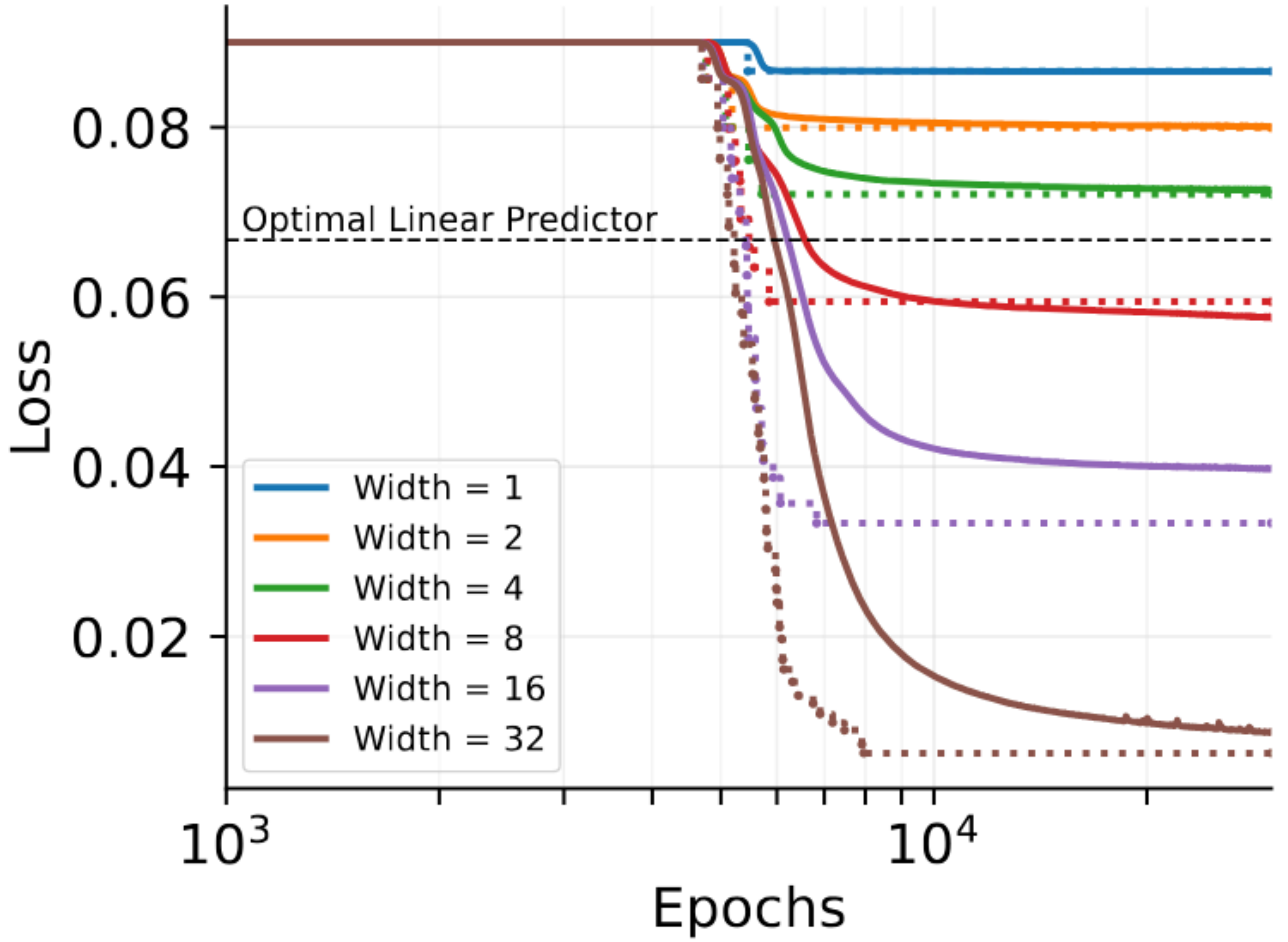
Function Space Perspective



Limitations of AGF and directions for future work

- On **natural data tasks**, loss curves are often not visibly stepwise even at very small initialization scales
  - Cost min steps bleed together
  - Many small steps lead to a smooth decay (emergent behavior and scaling laws)

Two-layer ReLU network on a subset of CIFAR-10



- AGF is an **ansatz for gradient flow** — can we prove a general conjecture (over a class of problems)

- Can we connect AGF to more classic feature learning analysis in **multi-index** models and **teacher–student** settings

- Can we extend the ideas of AGF to **deeper networks**? What is a “neuron”?

Thank you!

kunin@berkeley.edu

Poster Session: Thurs. Dec. 4th 11am-2pm

Exhibit Hall C,D,E, San Diego, CA