

Saddle-to-Saddle Dynamics Explains A Simplicity Bias Across Neural Network Architectures

Yedi Zhang¹, Andrew Saxe^{1,2}, and Peter E. Latham¹

¹Gatsby Computational Neuroscience Unit, University College London

²Sainsbury Wellcome Centre, University College London

Abstract

Neural networks trained with gradient descent often learn solutions of increasing complexity over time, a phenomenon known as simplicity bias. Despite being widely observed across architectures, existing theoretical treatments lack a unifying framework. We present a theoretical framework that explains a simplicity bias arising from saddle-to-saddle learning dynamics for a general class of neural networks, incorporating fully-connected, convolutional, and attention-based architectures. Here, *simple* means expressible with few hidden units, i.e., hidden neurons, convolutional kernels, or attention heads. Specifically, we show that linear networks learn solutions of increasing rank, ReLU networks learn solutions with an increasing number of kinks, convolutional networks learn solutions with an increasing number of convolutional kernels, and self-attention models learn solutions with an increasing number of attention heads. By analyzing fixed points, invariant manifolds, and dynamics of gradient descent learning, we show that saddle-to-saddle dynamics operates by iteratively evolving near an invariant manifold, approaching a saddle, and switching to another invariant manifold. Our analysis also illuminates the effects of data distribution and weight initialization on the duration and number of plateaus in learning, dissociating previously confounding factors. Overall, our theory offers a framework for understanding when and why gradient descent progressively learns increasingly complex solutions.

1 Introduction

Deep neural networks trained with gradient descent often learn functions of increasing complexity over the course of training (Arpit et al., 2017; Kalimeris et al., 2019; Rahaman et al., 2019; Saxe et al., 2019; Refinetti et al., 2023; Bhattacharya et al., 2023; Abbe et al., 2023). This dynamical simplicity bias has been observed across architectures (Shah et al., 2020; Teney et al., 2022; Rende et al., 2024), tasks (Wurgauf et al., 2025), and training paradigms ranging from supervised (Rahaman et al., 2019) to reinforcement (Schaul et al., 2019) and self-supervised learning (Simon et al., 2023). A particularly striking manifestation is stage-like dynamics: extended plateaus in loss alternating with bursts of rapid improvement as networks progress through increasingly complex input-output maps (Saxe et al., 2014; 2019). These dynamics, known as “saddle-to-saddle” dynamics because they can result from trajectories passing near a sequence of saddle points (Jacot et al., 2022; Berthier, 2023; Pesme & Flammarion, 2023), have been documented in deep linear networks (Saxe et al., 2014; 2019; Jacot et al., 2022), two-layer and deep ReLU networks (Maennel et al., 2018; Boursier et al., 2022; Chistikov et al., 2023; Wang & Ma, 2023; Kumar & Haupt, 2024; Zhang et al., 2025a; Wu et al., 2025; Bantzi et al., 2025), and self-attention models (Boix-Adsera et al., 2023; Geshkovski et al., 2024; Zhang et al., 2025b), and have been hypothesized to be universal (Ziyin et al., 2025; Kunin et al., 2025). Yet the same architectures can also exhibit smooth, exponential training dynamics, simply by changing the initialization (Jacot et al., 2018; Tu et al., 2024; Kunin et al., 2024); and more broadly, the emergence of stage-like dynamics can hinge on the data distribution (Yoshida & Okada, 2019; Goldt et al., 2020) and architectural choices (Orhan & Pitkow, 2018).

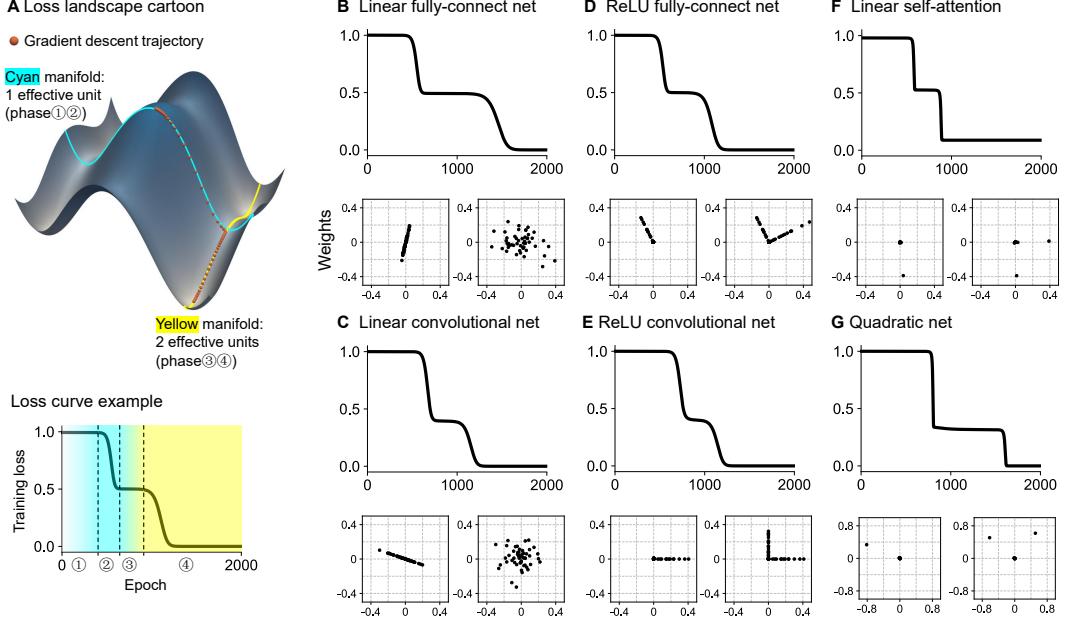


Figure 1: Saddle-to-saddle dynamics occurs in the gradient descent training of a wide range of architectures and leads to a dynamical simplicity bias. (A) Saddle-to-saddle dynamics on a cartoon loss landscape. The cyan and yellow curves represent invariant manifolds, on which the network implements input-output maps expressible by the architecture with one and two units, respectively. In general, saddle-to-saddle dynamics operates by repeating: i) during the plateau, escaping from a saddle associated with a width- h network onto an invariant manifold with effective width ($h + 1$); ii) during the rapid transition phase, approaching a fixed point on that manifold, which is a saddle associated with a width- $(h + 1)$ network. This figure shows two repeats of this process. (B-G) Loss and weight dynamics for various architectures. Each panel shows the loss during training (top), and the first-layer weights during the intermediate plateau (bottom left, phase 3 in panel A) and at the end of learning (bottom right). The first-layer weights to each hidden unit are two-dimensional and plotted as black dots. During the intermediate plateau, all networks visit a saddle, at which the input-output map of the network can be expressed by the architecture with only one unit. The network then converges to a stable fixed point, at which the input-output map is expressible with two units. The weight structures in BC, DE, and FG correspond to three categories of weight configurations of fixed points in Theorem 1; see Section 3 for details. A video version of this figure is provided at [URL](#). Dynamics with other two-layer architectures and deep networks are provided in Figures 3 to 5. Experimental details are provided in Section I.

These diverse findings raise foundational questions about the nature of dynamical simplicity bias in deep neural networks. Is there a universal mechanism driving stage-like dynamics, or a collection of architecture-specific mechanisms? Is there a principled link between stages and simplicity, such that earlier stages in training are simpler? And if simplicity does underlie these dynamics, what is the operative notion of simplicity, and how does it reflect an architecture’s inductive bias?

Here we answer these questions. We show that for a range of architectures, including linear networks, ReLU networks, convolutional networks, quadratic networks, and linear self-attention (Figure 1B-G), there is a universal mechanism, saddle-to-saddle dynamics, driving stage-like learning, and that there is a principled link between stages and a well-defined notion of simplicity. In particular, first we show that fixed points in the loss landscape are recursively embedded: fixed points of smaller networks are embedded in saddle points of larger networks, yielding a nested hierarchy of saddles. Second, we show that saddle points are connected by invariant manifolds along which a larger network behaves like a smaller one, preserving simplicity along the connecting trajectories. Third, the link between saddle-to-saddle dynamics and simplicity arises jointly from the recursive embedding of fixed points and a timescale separation which steers dynamics toward invariant manifolds associated with simple input-output maps. Ultimately, our results reveal that the relevant

notion of simplicity is the number of effective units in the architecture, i.e., hidden neurons, convolutional kernels, or attention heads. Concretely, a simpler solution is a solution with a lower rank for linear networks, fewer kinks for ReLU networks, fewer convolutional kernels for convolutional networks, and fewer heads for attention-based models. Together, this analysis paints a unified picture of embedded saddles, invariant manifolds, and dynamics which give rise to a dynamical simplicity bias across architectures, and predicts when instead non-stage-like behavior will arise.

Related work. We are inspired by a line of pioneering research that began with the seminal work of Fukumizu & Amari (2000) and was subsequently developed in follow-up studies (Inoue et al., 2003; Amari et al., 2006; Wei et al., 2008; Amari et al., 2011; Fukumizu et al., 2019; Simsek et al., 2021; Zhang et al., 2021). In particular, Fukumizu & Amari (2000) first discovered a hierarchy of fixed points in two-layer fully-connected nonlinear neural networks. While their fixed points could, in principle, be extended to convolutional and attention-based architectures, they did not explore this, as convolutional architectures had not been popularized and attention-based architectures had not been invented. We study the fixed points across fully-connected, convolutional, and attention-based architectures. Further, we go beyond fixed points to study invariant manifolds and saddle-to-saddle dynamics, with implications for simplicity bias. A more detailed discussion of related work is provided in Section A.

2 Network Setup

Let $f(\mathbf{x})$ represent a neural network with input $\mathbf{x} \in \mathbb{R}^D$. We focus on one layer in the network with H units and trainable parameters $\theta_{1:H}$,

$$f(\mathbf{x}; \theta_{1:H}) = g_{\text{out}} \left(\sum_{i=1}^H \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i) \mathbf{v}_i \right), \quad \text{where } \theta_i = \begin{bmatrix} \mathbf{v}_i \\ \mathbf{u}_i \end{bmatrix}. \quad (1)$$

Here $g_{\text{out}}(\cdot)$ and $g_{\text{in}}(\cdot)$ represent the processing after and before this layer, which are usually deeper and shallower layers of the network. The weights are $\mathbf{u}_i \in \mathbb{R}^{N_u}$, $\mathbf{v}_i \in \mathbb{R}^{N_v}$, and thus $\theta_i \in \mathbb{R}^{N_u + N_v}$. We place the second-layer weight \mathbf{v}_i on the right because $\phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i)$ may be a scalar (as in a fully-connected layer) or matrix (as in a self-attention layer). The network output $f(\mathbf{x}; \theta_{1:H})$ can be a scalar or vector. We will specify their dimensionality when we make them concrete.

The definition of a layer in Equation (1) incorporates major architectures. For a fully-connected layer, a unit is a hidden neuron: $\phi(z; \mathbf{w}, b) = \sigma(\mathbf{w}^\top z + b)$ where $\sigma(\cdot)$ is the activation function and \mathbf{w}, b are the weight and bias. For a convolutional layer, a unit is a convolutional kernel: $\phi(z; \mathbf{u}) = \sigma(\mathbf{u} * z)$ where $*$ denotes convolution. For a self-attention layer, a unit is an attention head: $\phi(\mathbf{Z}; \mathbf{K}, \mathbf{Q}) = \mathbf{I} \otimes \text{smax}(\mathbf{Z} \mathbf{Q} \mathbf{K}^\top \mathbf{Z}^\top) \mathbf{Z}$ where $\text{smax}(\cdot)$ denotes row-wise softmax and \mathbf{K}, \mathbf{Q} are the key and query weights. A self-attention layer fits into our definition as follows,

$$\text{ATTN}(\mathbf{Z}) = \text{smax}(\mathbf{Z} \mathbf{Q} \mathbf{K}^\top \mathbf{Z}^\top) \mathbf{Z} \mathbf{V} = \mathbf{I} \otimes \text{smax}(\mathbf{Z} \mathbf{Q} \mathbf{K}^\top \mathbf{Z}^\top) \text{Zvec}(\mathbf{V}) = \phi(\mathbf{Z}; \mathbf{K}, \mathbf{Q}) \mathbf{v}. \quad (2)$$

We note that this is not a common notation for self-attention; we present it solely to show that Equation (1) incorporates self-attention. Hence, statements we will make about Equation (1) apply to fully-connected, convolutional, and self-attention architectures.

Let $\{\mathbf{x}_\mu, \mathbf{y}_\mu\}_{\mu=1}^P$ be a supervised learning training set. The training loss is averaged over the training set $\mathcal{L} = \frac{1}{P} \sum_{\mu=1}^P \ell(\mathbf{y}_\mu, f(\mathbf{x}_\mu))$, where the loss function ℓ is second order differentiable with respect to $f(\mathbf{x})$, including common choices like squared error loss. The parameters are trained with gradient flow on the training loss,

$$\dot{\theta} = -\frac{\partial \mathcal{L}}{\partial \theta} = -\frac{\partial \mathcal{L}}{\partial f(\mathbf{x})} \frac{\partial f(\mathbf{x})}{\partial \theta}. \quad (3)$$

Gradient flow captures the behavior of gradient descent in the limit of a small learning rate.

Definition 1. A point θ^* is a fixed point of the gradient flow dynamics in Equation (3) if $\frac{\partial \mathcal{L}}{\partial \theta}|_{\theta^*} = \mathbf{0}$.

3 Loss Landscape: Embedded Fixed Points

In this section, we establish that saddles generally exist in networks described by Equation (1). We show that a fixed point of a narrow network gives rise to a set of fixed points in a wider network.

These fixed points are constructed by embedding the narrow network into the wider network, as formalized in Theorem 1.

Theorem 1 (Embedded fixed points). *If a network defined by Equation (1) with $(H - 1)$ units has a fixed point $\theta_{1:(H-1)}^*$ yielding an input-output map $f^*(\mathbf{x})$, then there exists $\theta_{1:H} \in \mathcal{S}$ such that a network with H units implements the same map $f^*(\mathbf{x})$ and $\theta_{1:H}$ is a fixed point.*

We construct $\theta_{1:H}$ by setting the first $(H - 1)$ units to $\theta_{1:(H-1)}^*$ and modifying them as follows.

(i) For any ϕ , the set \mathcal{S} includes

$$\mathbf{u}_H = \mathbf{u}_i^*, \mathbf{v}_H = \gamma_v \mathbf{v}_i^*, \mathbf{v}_i = (1 - \gamma_v) \mathbf{v}_i^*, \quad \gamma_v \in \mathbb{R}, i \in \{1, \dots, H - 1\}. \quad (4)$$

(ii) If $\exists \mathbf{u}_{\text{zero}}$ such that $\forall \mathbf{z}, \phi(\mathbf{z}; \mathbf{u}_{\text{zero}}) = 0$, the set \mathcal{S} includes

$$\mathbf{u}_H = \mathbf{u}_{\text{zero}}, \mathbf{v}_H = \mathbf{0}. \quad (5)$$

(iii) If $\phi(\mathbf{z}; \mathbf{u})$ is degree-1 homogeneous in \mathbf{u} , that is $\forall \alpha \in \mathbb{F}, \phi(\mathbf{z}; \alpha \mathbf{u}) = \alpha \phi(\mathbf{z}; \mathbf{u})$, where $\mathbb{F} = \mathbb{R}$ for general homogeneous functions, and $\mathbb{F} = \mathbb{R}_{\geq 0}$ for positively homogeneous functions, e.g., the ReLU activation function, the set \mathcal{S} includes

$$\mathbf{u}_H = \gamma_u \mathbf{u}_i^*, \mathbf{v}_H = \gamma_v \mathbf{v}_i^*, \mathbf{v}_i = (1 - \gamma_u \gamma_v) \mathbf{v}_i^*, \quad \gamma_v \in \mathbb{R}, \gamma_u \in \mathbb{F}, i \in \{1, \dots, H - 1\}. \quad (6)$$

(iv) If $\phi(\mathbf{z}; \mathbf{u})$ is linear in \mathbf{u} , that is degree-1 homogeneous, $\forall \alpha \in \mathbb{R}, \phi(\mathbf{z}; \alpha \mathbf{u}) = \alpha \phi(\mathbf{z}; \mathbf{u})$, and additive, $\phi(\mathbf{z}; \mathbf{u}_i) + \phi(\mathbf{z}; \mathbf{u}_j) = \phi(\mathbf{z}; \mathbf{u}_i + \mathbf{u}_j)$, the set \mathcal{S} includes

$$\begin{aligned} \mathbf{u}_H &= \sum_{i=1}^{H-1} \gamma_{u_i} \mathbf{u}_i^*, \mathbf{v}_H = \sum_{i=1}^{H-1} \gamma_{v_i} \mathbf{v}_i^*, \quad \gamma_{v_i}, \gamma_{u_i} \in \mathbb{R}, \\ \mathbf{v}_i &= \mathbf{v}_i^* - \gamma_{u_i} \sum_{j=1}^{H-1} \gamma_{v_j} \mathbf{v}_j^*, \quad i = 1, \dots, H - 1. \end{aligned} \quad (7)$$

The proof of Theorem 1, which is provided in the Section E, consists of two steps. First, verify that for the weight configurations given above, the width- H network implements the same input-output map as the width- $(H - 1)$ network. Second, show that gradients of the weights in the width- H network are either equal or proportional to those in the width- $(H - 1)$ network, which are zero.

Remark 1. Equation (4) is valid for any activation function ϕ , while the rest are valid for ϕ with specific properties, implying that certain properties of ϕ give rise to a larger set of embedded fixed points in weight space. Equations (4) and (5) were first discovered by Fukumizu & Amari (2000). We extend these two constructions with Equations (6) and (7). This extension is crucial for studying learning dynamics, as the saddles visited during learning turn out to fall under Equations (5) to (7) but not Equation (4).

By induction, we obtain Corollary 2 by repeatedly applying Theorem 1 to embed multiple units in one layer and embed units in multiple layers of a deep network, with each layer defined by Equation (1).

Corollary 2. *If a depth- L network with h_l units in layer l ($l = 1, \dots, L$) has a fixed point yielding an input-output map $f^*(\mathbf{x})$, then for a depth- L network with $H_l \geq h_l$ units in each layer, there exist weight configurations such that the network implements the same map $f^*(\mathbf{x})$ and the weight configurations are fixed points.*

Theorems 1 and 2 indicate that the global minima of a narrow network, even if they incur nonzero training loss, remain fixed points of the gradient flow dynamics in any wider network with the same architecture. For example, the global minimum of a width-1 network typically lacks the expressivity to fit the training set and thus incurs nonzero loss. In a wide network capable of achieving zero loss, the fixed points corresponding to the width-1 network global minimum are either saddles or local minima. They are guaranteed to be saddles in deep linear networks with rank- r ($r \geq 1$) target maps (Baldi & Hornik, 1989; Kawaguchi, 2016) and, under mild conditions, are saddles in general architectures (Fukumizu & Amari, 2000; Fukumizu et al., 2019).

In Figure 1, we show six cases where the network first visits a saddle, corresponding to a solution expressible by the architecture with a single unit. The network then converges to a stable fixed point,

corresponding to a solution expressible with two units. The fixed points visited during learning fit into three different categories in Theorem 1. In panels (B,C), the fixed points visited during learning are described by Equation (7), corresponding to rank-one and rank-two weights. In panels (D,E), the fixed points are described by Equation (6), corresponding to one and two rays of proportional weights. In panels (E,F), the fixed points are described by Equation (5), corresponding one or two units with large weights with the rest being near zero.

4 Invariant Manifold: Effectively Narrow Networks

An invariant manifold of a dynamical system is a manifold such that any point starting on it remains on the manifold under the system's evolution. In Theorem 3, we show that for gradient flow dynamics of the class of neural networks we consider, invariant manifolds always exist. Further, these invariant manifolds correspond to weight configurations that make the network effectively narrower than its actual width.

Theorem 3 (Invariant manifolds). *Let T be any time such that one of the following conditions (i)-(iv) holds in a network defined by Equation (1). Then, in each case, the stated relationship between the weights is preserved for all $t \geq T$ under gradient flow dynamics:*

- (i) *For any ϕ , two units have equal weights: $\theta_i = \theta_j$.*
- (ii) *If $\exists \mathbf{u}_{\text{zero}}$ such that $\forall z, \phi(z; \mathbf{u}_{\text{zero}}) = 0$, a unit has zero weights: $\mathbf{v}_i = \mathbf{0}, \mathbf{u}_i = \mathbf{u}_{\text{zero}}$.*
- (iii) *If $\phi(z; \mathbf{u})$ is homogeneous in \mathbf{u} , two units have proportional weights: $\theta_i = \gamma \theta_j, \gamma \in \mathbb{F}$.*
- (iv) *If $\phi(z; \mathbf{u})$ is linear in \mathbf{u} , any number of units have linear dependence: $\theta_i = \sum_{j \neq i} \gamma_j \theta_j$.*

The precise definitions of homogeneity and linearity are given in Theorem 1.

The proof of Theorem 3 is provided in the Section F and is relatively straightforward. For example, when $\theta_i = \theta_j$, the gradients of θ_i and θ_j are equal and thus they stay equal for all future time. The invariant manifolds are larger in weight space when ϕ has zero, homogeneity or linearity properties, similar to the enlarged set of embedded fixed points in Theorem 1.

When the weights of a network lie on an invariant manifold, its input-output map is expressible with fewer units than its actual width: simply remove the i -th unit and appropriately modify the remaining weights (see Section F.3). Further, we can have more than one constraints; e.g., $\theta_1 = \theta_2$ and $\theta_3 = \theta_4$. Each added constraint reduces the effective width by 1. Hence, when weights evolve on an invariant manifold, the simplicity of the network's input-output map is constrained by the effective width associated with the invariant manifold, rather than the actual width.

The invariant manifolds indicate that there exist gradient flow paths connecting pairs of embedded fixed points defined in Theorem 1 (see Section F.4). Following such a path corresponds to an iteration of saddle-to-saddle dynamics. To see this, starting from an embedded fixed point with effective width h , we may apply a carefully chosen small perturbation that moves the weights onto the invariant manifold with effective width $(h + 1)$. This perturbation corresponds to breaking exactly one constraint. By Theorem 3, the dynamics then remains on the invariant manifold for all time, eventually converging to a fixed point on it, that is, an embedded fixed point with effective width $(h + 1)$. This process is one saddle-to-saddle transition: from the saddle with effective width h to the saddle with $(h + 1)$. We illustrate this process in Figure 1A. In the next section, we develop heuristic arguments showing that the gradient flow dynamics can, in some cases, naturally evolve near such saddle-to-saddle paths on the invariant manifolds.

5 Saddle-to-Saddle Dynamics

The embedded fixed points (Section 3) and invariant manifolds (Section 4) hold for general architectures defined by Equation (1). To analyze learning dynamics, however, we must work with concrete architectures. We focus on two-layer networks where $\phi(\mathbf{x}; \mathbf{u})$ is a homogeneous polynomial in the weights \mathbf{u} , studying the linear and quadratic cases in detail. The linear case includes fully-connected linear networks and convolutional linear networks. The quadratic case includes quadratic networks (defined by Equation (71)) and linear self-attention. Both types of architectures exhibit saddle-to-saddle dynamics, but their mechanisms differ. We show that the mechanism in the linear case is a

timescale separation between directions across all units due to the distribution of the data, while the mechanism in the quadratic case is a timescale separation between units due to initialization.

5.1 Linear case: timescale separation between directions

Consider a two-layer network in which $\phi(\mathbf{x}; \mathbf{u})$ is linear in the weights \mathbf{u} ,

$$f(\mathbf{x}; \boldsymbol{\theta}_{1:H}) = \sum_{i=1}^H \mathbf{v}_i \mathbf{u}_i^\top \mathbf{z}(\mathbf{x}) \equiv \mathbf{W} \mathbf{z}, \quad \text{where } \mathbf{v} \in \mathbb{R}^{N_v}, \mathbf{u}, \mathbf{z} \in \mathbb{R}^{N_u}. \quad (8)$$

Here $\mathbf{z}(\mathbf{x})$ denotes any function of the input \mathbf{x} , as $\phi(\mathbf{x}; \mathbf{u})$ is linear in \mathbf{u} but not necessarily linear in \mathbf{x} . The gradient flow dynamics of Equation (8) trained on squared loss is

$$\dot{\mathbf{v}}_i = (\Sigma_{yz} - \mathbf{W} \Sigma_{zz}) \mathbf{u}_i, \quad \dot{\mathbf{u}}_i = (\Sigma_{yz} - \mathbf{W} \Sigma_{zz})^\top \mathbf{v}_i, \quad i = 1, \dots, H, \quad (9)$$

where the data statistics are $\Sigma_{yz} = \frac{1}{P} \sum_{\mu=1}^P \mathbf{y}_\mu \mathbf{z}_\mu^\top$, $\Sigma_{zz} = \frac{1}{P} \sum_{\mu=1}^P \mathbf{z}_\mu \mathbf{z}_\mu^\top$. When the weights are initialized to be small, i.e., $\mathbf{v}_i(0) = O(\epsilon)$, $\mathbf{u}_i(0) = O(\epsilon)$, $i = 1, \dots, H$, the first terms in Equation (9) dominate: $\Sigma_{yz} - \mathbf{W} \Sigma_{zz} = \Sigma_{yz} + O(\epsilon^2)$. The weights thus approximately evolve as a linear dynamical system (Equation (10)), which we analyze in Theorem 4.

Theorem 4 (Timescale separation between directions). *Consider the linear dynamical system*

$$\dot{\mathbf{v}}_i = \Sigma_{yz} \mathbf{u}_i, \quad \dot{\mathbf{u}}_i = \Sigma_{yz}^\top \mathbf{v}_i, \quad i = 1, \dots, H. \quad (10)$$

Let the singular value decomposition of Σ_{yz} be given by $\Sigma_{yz} = \sum_{k=1}^D s_k \mathbf{q}_k \mathbf{r}_k^\top$, $D = \min(N_v, N_u)$ with singular values $s_1 \geq \dots \geq s_D$, and let the largest singular value s_1 have multiplicity r ($1 \leq r < D$). Let the initial weights be sampled independently from a Gaussian distribution $\mathcal{N}(0, \epsilon^2)$ with a small ϵ . When the projection of the weights on the span of the top r singular vectors reaches $O(1)$, that is

$$\|\mathbf{P} \boldsymbol{\theta}_i\| = O(1), \quad \text{where } \mathbf{P} = \frac{1}{2} \sum_{k=1}^r \begin{bmatrix} \mathbf{q}_k \\ \mathbf{r}_k \end{bmatrix} \begin{bmatrix} \mathbf{q}_k^\top & \mathbf{r}_k^\top \end{bmatrix}, \quad \boldsymbol{\theta}_i = \begin{bmatrix} \mathbf{v}_i \\ \mathbf{u}_i \end{bmatrix}, \quad (11)$$

the projection on the remaining subspace is $\|(I - \mathbf{P}) \boldsymbol{\theta}_i\| = O(\epsilon^{1-s_{r+1}/s_1})$ almost surely.

We provide the proof in Section G.2 and the intuition here. The second and first-layer weights $\mathbf{v}_i, \mathbf{u}_i$ grow exponentially along the singular vectors $\mathbf{q}_k, \mathbf{r}_k$, respectively, at the rate $e^{s_k t}$. Relative to the dominant growth rate $e^{s_1 t}$ along the top singular vectors, the components along other singular vectors decay as $e^{(s_k - s_1)t}$, $k = r + 1, \dots, D$. Consequently, during the early phase, the weights become increasingly aligned with the top singular vectors and thus approximately rank- r . Taking $r = 1$ as an example, the weights become approximately rank-one; specifically, \mathbf{v}_i aligns with \mathbf{q}_1 , and \mathbf{u}_i aligns with \mathbf{r}_1 for every i .

Theorem 3 implies that rank- r weights constrain a linear network to an invariant manifold corresponding to effective width r . Since the early phase dynamics drives the weights to be approximately rank- r , the network evolves near the invariant manifold and approaches a fixed point on it. This is the first iteration of saddle-to-saddle dynamics. In weight space, the weights move from the initial saddle at zero to the second saddle. In function space, the network learns a more complex solution, changing from a constant zero function to a rank- r projection of the target linear map.

Subsequent iterations of saddle-to-saddle dynamics operate similarly. The dynamics near a rank- r saddle, corresponding to a plateau in the loss, is again approximately a linear dynamical system

$$\dot{\mathbf{v}}_i = \tilde{\Sigma}_{yz} \mathbf{u}_i, \quad \dot{\mathbf{u}}_i = \tilde{\Sigma}_{yz}^\top \mathbf{v}_i, \quad i = 1, \dots, H. \quad (12)$$

where $\tilde{\Sigma}_{yz}$ is Σ_{yz} projected onto a rank- $(D - r)$ subspace; see Section G.3. Via the same reasoning as Theorem 4, the weights grow the fastest along the top singular vectors of $\tilde{\Sigma}_{yz}$. Low-rank weight growth will again place a linear network near an invariant manifold with few more effective units, guiding the dynamics toward a fix point on that manifold.

To summarize, in the linear case, distinct singular values of the input-output correlation matrix induce a timescale separation between weight growth along different directions. If all singular values are distinct, the timescale separation leads to approximately rank-one weight growth during a loss plateau, causing the escape path from a saddle to closely follow an invariant manifold with one more effective unit.

5.2 Quadratic case: timescale separation between units

We now consider a two-layer network in which $\phi(\mathbf{x}; \mathbf{u})$ is quadratic in the weights \mathbf{u} ,

$$f(\mathbf{x}; \boldsymbol{\theta}_{1:H}) = \sum_{i=1}^H v_i \mathbf{u}_i^\top \mathbf{Z}(\mathbf{x}) \mathbf{u}_i, \quad \text{where } v_i \in \mathbb{R}, \mathbf{u}_i \in \mathbb{R}^D, \mathbf{Z} \in \mathbb{R}^{D \times D}. \quad (13)$$

Here $\mathbf{Z}(\mathbf{x})$ denotes any function of the input \mathbf{x} . For example, linear self-attention fits into Equation (13) with $\mathbf{Z}(\mathbf{x})$ being a cubic function of the input \mathbf{x} , and $\phi(\mathbf{x}; \mathbf{u})$ a quadratic function of the key and query weights $\mathbf{u} = [\text{vec}(\mathbf{K}), \text{vec}(\mathbf{Q})]$. We consider the scalar output case because it already has saddle-to-saddle dynamics and involves non-closed-form solutions. The gradient flow dynamics of Equation (13) trained on squared loss is given by Equation (44). Near small initialization, the quadratic terms in Equation (44) dominate. In Theorem 5, we analyze the approximate dynamics and show that one unit with the largest initialization grows much faster than the rest.

Proposition 5 (Timescale separation between units). *Consider the dynamical system*

$$\dot{v}_i = \mathbf{u}_i^\top \boldsymbol{\Sigma}_{yZ} \mathbf{u}_i, \quad \dot{\mathbf{u}}_i = 2v_i \boldsymbol{\Sigma}_{yZ} \mathbf{u}_i, \quad i = 1, \dots, H. \quad (14)$$

Assume $\boldsymbol{\Sigma}_{yZ}$ is symmetric and has both positive and negative eigenvalues. Let the initial weights be sampled independently from a Gaussian distribution $\mathcal{N}(0, \epsilon^2)$ with a small ϵ . When weights in one of the units reaches $O(1)$, the rest of the units is $O(\epsilon)$ almost surely.

We provide derivations in Section H.2 and the intuition here. The intuition is that the quadratic dynamics in Equation (14) is a rich-get-richer process. We can get a flavor of such dynamics by considering the simplest quadratic dynamics, $\dot{v}_i = v_i^2$, which has the solution

$$v_i(t) = \left(\frac{1}{v_i(0)} - t \right)^{-1}, \quad i = 1, \dots, H. \quad (15)$$

By solving for t with i and j , we can write $v_i(t)$ in terms of $v_j(t)$ as

$$v_i(t) = \left[\frac{1}{v_j(0)} \left(\frac{v_j(0)}{v_i(0)} - 1 \right) + \frac{1}{v_j(t)} \right]^{-1}. \quad (16)$$

Assuming initial conditions of order $O(\epsilon)$, for example $v_i(0) \sim \mathcal{N}(0, \epsilon^2)$, and letting v_j be the unit with the largest initial value, we see that when $v_j(t) \sim O(1)$, the other units are still small: $v_i(t) \sim O(\epsilon)$ for $i \neq j$. Thus, under quadratic dynamics $\dot{v}_i = v_i^2$, distinct initial conditions of the units induce a timescale separation in their growth. Although the general case, analyzed in Section H.2, is more complicated, the timescale separation between units essentially comes from the same mechanism.

In Theorem 3(ii), we showed that if $\phi(\mathbf{x}; \mathbf{0}) = 0 \forall \mathbf{x}$, then nonzero weights in one unit and zero weights in the rest of the units constrain a network to an invariant manifold with effective width one. Since the early dynamics drives one unit to grow much faster than the rest, the network evolves near the invariant manifold with effective width one and approaches a fixed point on it. This is the first iteration of saddle-to-saddle dynamics. Subsequent iterations operate similarly. Starting near the first saddle, one unit has nonzero weights and $(H - 1)$ units still have small weights. The dynamics near the first saddle drives one of the $(H - 1)$ units to grow much faster than the rest. Hence, the escape path from the first saddle again approximately follow the invariant manifold with two effective units, steering the dynamics toward a fixed point on that manifold. This process repeats.

For $\phi(\mathbf{x}; \mathbf{u})$ that is quadratic in \mathbf{u} and has $\phi(\mathbf{x}; \mathbf{0}) = 0 \forall \mathbf{x}$, the distinct initial weights in each unit induce a timescale separation between the weight growth in different units. One unit grows much faster than the rest, causing the escape path from a saddle to closely follow an invariant manifold with one more effective unit.

Higher-order polynomial activation. If $\phi(\mathbf{x}; \mathbf{u})$ is a homogeneous polynomial of degree $p > 2$ in the weights \mathbf{u} , we conjecture that there is still a timescale separation between units, possibly even stronger than the quadratic case. Our intuition is that the dynamics near zero has a similar flavor to the scalar dynamics, $\dot{v}_i = v_i^p$. By similar reasoning to Theorem 5, the unit with the largest initialization grows much faster than the rest, causing a timescale separation between units. The dynamics in the cubic ($p = 3$) case is consistent with our intuition, as shown in Figure 4G.

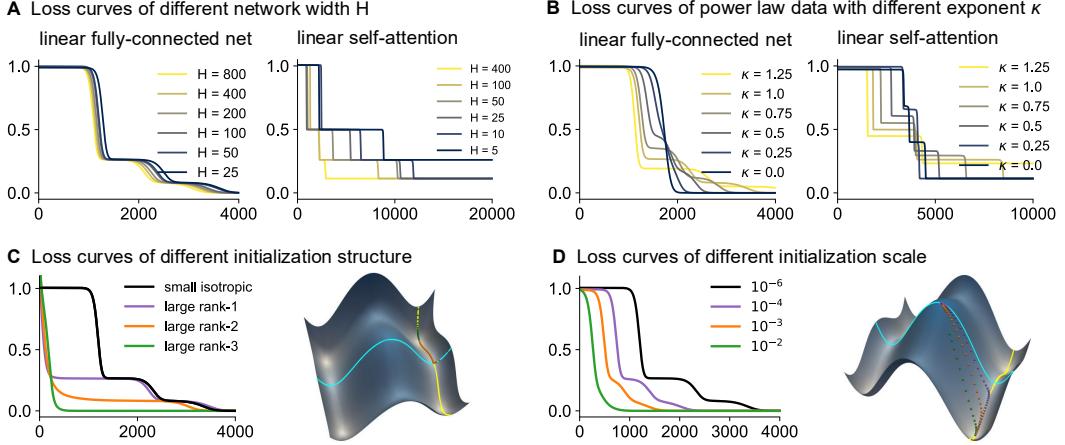


Figure 2: The effect of network width, data distribution, and initialization on learning dynamics. Singular values of Σ_{yz} (linear network) or positive singular values of Σ_{yZ} (linear self-attention) follow a power law, $s_n = n^{-\kappa}$, $n = 1, 2, 3$, and are normalized such that $\sum_{n=1}^3 s_n = 1$. (A) Increasing the number of units H has little effect on the loss curves of linear networks, but shortens the plateaus in linear self-attention. $\kappa = 1$ for both models. (B) Decreasing the power law exponent κ shortens the plateaus in both linear networks and linear self-attention. Setting $\kappa = 0$ eliminates plateaus in linear networks but does not eliminate plateaus in linear self-attention. H is 100 for linear networks and 25 for linear self-attention. (C) Linear networks with small isotropic initialization or large low-rank initialization exhibit saddle-to-saddle dynamics. The loss landscape cartoon illustrates large rank- r weights places a linear network near an invariant manifold with r effective units and thus approaches saddles during learning. (D) Increasing the scale of isotropic random initialization shortens the plateaus. $\kappa = 1$ for panels C,D.

General nonlinear activation. If $\phi(\mathbf{x}; \mathbf{u})$ is a general nonlinear activation function, we can Taylor expand $\phi(\mathbf{x}; \mathbf{u})$ around $\mathbf{u} = \mathbf{0}$. With small initialization, $\mathbf{u} \approx \mathbf{0}$, the early dynamics near initialization is dominated by the lowest-order non-vanishing term in the Taylor expansion, assuming the data statistic associated with that term is nonzero. For example, in a two-layer fully-connected tanh network, the lowest-order non-vanishing term is the linear term. The tanh network thus develops rank-one weights in the early phase near initialization, similar to Theorem 4. However, the subsequent dynamics is not necessarily saddle-to-saddle, since rank-one weights do not generally correspond to invariant manifolds for tanh networks; see Figure 4D. By comparison, in a two-layer fully-connected network with activation $\phi(\mathbf{x}; \mathbf{u}) = \mathbf{u}^\top \mathbf{x} \cdot \tanh(\mathbf{u}^\top \mathbf{x})$, the lowest-order non-vanishing term is quadratic. The network thus has a timescale separation between units similar to Theorem 5, and exhibits saddle-to-saddle dynamics as shown in Figure 4F.

6 Implications

We now validate our theory and demonstrate its predictive power by examining how the network width, data distribution, and initialization affect learning dynamics.

Effect of network width. Our analysis in Section 5.1 shows that in linear networks, the timescale separation occurs between directions across all units. Consequently, increasing the number of units in linear networks has little effect on the dynamics, provided there are enough units to learn all directions. In contrast, the analysis in Section 5.2 implies that increasing the number of units in networks where $\phi(\mathbf{x}; \mathbf{u})$ is quadratic in \mathbf{u} can shorten the plateaus. That is because the timescale separation in the quadratic case occurs between learning different units due to their distinct initial values. When sampling initial weights from a fixed distribution, increasing the number of weights reduces the gaps between adjacent samples, thereby shortening the plateaus. Simulations in Figure 2A confirm our theoretical prediction. In this case, increasing the number of heads of linear self-attention, for which $\phi(\mathbf{x}; \mathbf{u})$ is quadratic in \mathbf{u} , speeds up learning, while increasing the width of fully-connected linear networks does not. This demonstrates an interesting, theoretically grounded advantage of scaling up linear self-attention over scaling up fully-connected linear networks.

Effect of data distribution. In linear networks, the timescale separation in learning different directions arises from the distinct singular values of Σ_{yz} . In Figure 2B, we let the singular values of Σ_{yz} follow a power law. As expected, decreasing the power law exponent narrows the gaps between singular values, thereby shortening the plateaus. When the exponent is 0, all the singular values are equal, eliminating the plateaus except the initial one corresponding to the escape from the saddle at zero. In this case, the largest singular value has multiplicity $r = D$ in Theorem 4, causing the solution to jump directly from effective width 0 to D , skipping the stages in between. By contrast, in networks for which $\phi(\mathbf{x}; \mathbf{u})$ is quadratic in \mathbf{u} , the timescale separation is due to the distinct initial values in the units. Therefore, setting the positive singular values of Σ_{yz} to be equal shortens but does not eliminate plateaus. Simulations with linear self-attention in Figure 2B confirm our prediction.

Effect of initialization structure. According to our theory, to have saddle-to-saddle dynamics the initialization must be near an invariant manifold, and the escape path from saddles must follow an invariant manifold. Perhaps surprisingly, however, initializing near a saddle is not a necessary condition. In Figure 2C we initialize the weights near an invariant manifold but away from saddles; for linear networks, this corresponds to large low-rank weights with a small perturbation. As predicted, learning undergoes saddle-to-saddle dynamics. Because the initialization is away from saddles, there is not a plateau at the start; the loss first drops exponentially and then exhibits plateaus followed by sigmoid-shaped drops. To our knowledge, this regime has not previously been observed. If we initialize near the invariant manifold associated with exactly the required number of effective units, loss undergoes a rapid exponential drop, even though the network learns a solution with low-rank weights, which is the feature learning solution in linear networks (Dominé et al., 2025). This result adds nuance to the common view that exponential loss curves are often a hallmark of lazy learning (Jacot et al., 2018; Chizat et al., 2019).

Effect of initialization scale. We examine the effect of initialization scale when using an isotropic Gaussian distribution, a common choice in practice. As shown in Figure 2D, increasing the initialization scale gradually shortens the plateaus. Saddle-to-saddle dynamics becomes weaker in the sense that the learning trajectory does not approach the saddles as closely as it does with small initialization. For intermediate initialization scales, plateaus are less pronounced, yet the network still approximately learns solution of increasing complexity, similar to the case with small initialization. In architectures that have saddle-to-saddle dynamics, we conjecture that the distance from the initial weights to invariant manifolds associated with low effective width determines the strength of feature learning. This criterion can be viewed as an extension of prior beliefs, in which the relative scale of initial weights across layers (Dominé et al., 2025) or the rank of initial weights (Liu et al., 2024) were thought to determine the strength of feature learning.

7 Discussion

We studied the gradient flow dynamics of a broad class of architectures, analyzing fixed points, invariant manifolds, and dynamics near fixed points. Our theoretical framework reveals a general mechanism for saddle-to-saddle dynamics and provides a definition of simplicity that reflects the inductive biases of different architectures. When a network exhibits saddle-to-saddle dynamics, it recruits one or a few new effective units during each transition and learns solutions of increasingly complexity, where complexity is measured by the minimal number of units required for the architecture to express the solution. On a high level, we identify a mechanism behind the intuition that a neural network can decompose a task into smaller pieces and learn piece by piece over time. The learning process sometimes reconstructs the network’s own architecture, one unit at a time.

Condition for saddle-to-saddle dynamics. Saddle-to-saddle dynamics depends on two conditions: (i) the escape path from saddles closely follows invariant manifolds with few additional effective units; and (ii) the initialization is close to an invariant manifold with fewer effective units than needed to attain zero loss. As an example violating the first condition, two-layer tanh networks with small initialization develop rank-one weights during the early phase. This is because the tanh function is approximately linear near zero, and thus the early dynamics is approximately a linear dynamical system, similar to Theorem 4. However, since tanh is not homogeneous, rank-one weights do not correspond to an invariant manifold with effective width one. Consequently, tanh networks are not guided to approach the saddle with one effective unit, and probably do not have saddle-to-

saddle dynamics in general. As an example violating the second condition, large isotropic random initialization is almost surely away from invariant manifolds. Thus, neural networks with large random initialization generally do not exhibit saddle-to-saddle dynamics. A special case violating the second condition is when an architecture has full expressivity with a single unit, such as linear networks with scalar input or scalar output (Shamir, 2019), and linear self-attention with merged key and query weights (Zhang et al., 2025b).

Deep networks. The fixed points and invariant manifolds in Sections 3 and 4 apply to general deep networks defined by Equation (1), whereas the analysis of dynamics in Section 5 only applies to two-layer networks. Nonetheless, many deep networks still exhibit saddle-to-saddle dynamics, with some showing a timescale separation between directions and some between units, as shown in Figure 5. Although a general treatment of deep network dynamics is beyond the scope of this paper, we propose a conjecture for predicting which type of timescale separation (between directions or units) arises within a layer of a deep network. We conjecture that the order of the activation function $\phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i)$, whether it is linear or quadratic in \mathbf{u}_i , continues to predict learning behaviors, including the type of the timescale separation and the effects of width and data distribution. In deep networks, $g_{\text{in}}(\mathbf{x})$ in Equation (1) may involve weights that are not specific to any individual unit of the layer under consideration, i.e., weights in shallower layers not indexed by i . For example, let us consider the second hidden layer of a depth-3 linear fully-connected network:

$$f(\mathbf{x}) = \sum_{i=1}^H \mathbf{v}_i \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i) = \sum_{i=1}^H \mathbf{v}_i \mathbf{u}_i^\top g_{\text{in}}(\mathbf{x}), \quad \text{where } g_{\text{in}}(\mathbf{x}) = \mathbf{W}\mathbf{x}, \quad (17)$$

where \mathbf{W} is the first-layer weight matrix.¹ Since $\phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i) = \mathbf{u}_i^\top g_{\text{in}}(\mathbf{x})$ is linear in \mathbf{u}_i , we predict a timescale separation between directions similar to Section 5.1, and that the weights acquire an additional rank during each saddle-to-saddle transition. This is consistent with the existing literature (Gidel et al., 2019; Gissin et al., 2020) and our simulations in Figure 5.

We further note that deep networks introduce several new questions that do not arise in the two-layer setting. If deep networks visit a sequence of embedded fixed points and learn increasingly complex solutions by recruiting additional effective units, which layers recruit additional units at each increase in complexity? This question is particularly interesting for transformers, which have self-attention, fully-connected layers, and skip connections. With skip connections, a deep network may also learn increasingly complex solutions by recruiting additional layers. This possibility seems consistent with the literature on layer pruning showing that large-scale transformers maintain their performance when removing up to half of the deeper layers and performing a small amount of finetuning (Gromov et al., 2025). Another work modeled the increasingly complex solutions of a transformer by increasing the width of its fully-connected layers (Wurgaft et al., 2025).

Exhaustiveness of fixed points and invariant manifolds. Although we have not identified any fixed points or invariant manifolds beyond Theorems 3 and 5, it remains an open question whether these are exhaustive. If not, under what conditions do they become so? If the fixed points are exhaustive under reasonable assumptions, they would provide a useful diagnostic: each plateau during training would indicate that the network is implementing a solution expressible by a narrower sub-network. Moreover, the fixed points and invariant manifolds we describe arise solely from the network architecture and thus hold for any training data set. A further question is whether particular data sets can induce more fixed points or invariant manifolds than the data-agnostic ones (Zhao et al., 2023; Misof et al., 2025).

Other architectures and learning rules. At its core, our theory exploits the permutation symmetry of units in feed-forwards neural networks defined by Equation (1). Permutation symmetry exists beyond feed-forward architectures and supervised learning rules. Indeed, stage-like learning curves have been observed in recurrent neural networks (Proca et al., 2025; Ger & Barak, 2025), and other learning rules, such as reinforcement learning (Schaul et al., 2019), self-supervised learning (Simon et al., 2023), and predictive coding (Innocenti et al., 2024). This suggests the possibility of an even broader theory that incorporates these architectures and learning rules, with progressive permutation symmetry breaking as a unifying explanation for progressive learning behaviors.

¹A depth-3 linear network differs from linear self-attention, $\sum_{i=1}^H \mathbf{X} \mathbf{Q}_i \mathbf{K}_i^\top \mathbf{X}^\top \mathbf{X} \mathbf{V}_i$, because in linear self-attention all weights are indexed by i , and thus cannot be absorbed into $g_{\text{in}}(\mathbf{x})$.

Acknowledgments

We thank Samuel Liebana, Loek van Rossem, Erin Grant, Stefano Sarao Mannelli, Máté Lengyel, Valentina Njaradi, Aaditya K. Singh, Andrew Lampinen, and Jin Hwa Lee for helpful conversations, and anonymous reviewers for their constructive feedback.

We thank the following funding sources: Gatsby Charitable Foundation (GAT3850 and GAT4058) to YZ, AS, and PEL; Sainsbury Wellcome Centre Core Grant from Wellcome (219627/Z/19/Z) to AS; Schmidt Science Polymath Award to AS. AS is a CIFAR Azrieli Global Scholar in the Learning in Machines & Brains program.

References

- Emmanuel Abbe, Enric Boix Adserà, and Theodor Misiakiewicz. Sgd learning on neural networks: leap complexity and saddle-to-saddle dynamics. In Gergely Neu and Lorenzo Rosasco (eds.), *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pp. 2552–2623. PMLR, 12–15 Jul 2023. URL <https://proceedings.mlr.press/v195/abbe23a.html>.
- El Mehdi Achour, François Fleuret, Sébastien Gerchinovitz. The loss landscape of deep linear neural networks: a second-order analysis. *Journal of Machine Learning Research*, 25(242):1–76, 2024. URL <http://jmlr.org/papers/v25/23-0493.html>.
- Madhu S. Advani, Andrew M. Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020. ISSN 0893-6080. doi: <https://doi.org/10.1016/j.neunet.2020.08.022>. URL <https://www.sciencedirect.com/science/article/pii/S0893608020303117>.
- Shun-ichi Amari, Hyeyoung Park, and Tomoko Ozeki. Singularities affect dynamics of learning in neuromanifolds. *Neural Computation*, 18(5):1007–1065, 05 2006. ISSN 0899-7667. doi: 10.1162/neco.2006.18.5.1007. URL <https://doi.org/10.1162/neco.2006.18.5.1007>.
- Shun-ichi Amari, Tomoko Ozeki, Florent Rousseau, and Haikun Wei. Dynamics of learning in hierarchical models – singularity and milnor attractor. In Rubin Wang and Fanji Gu (eds.), *Advances in Cognitive Neurodynamics (II)*, pp. 3–9, Dordrecht, 2011. Springer Netherlands. ISBN 978-90-481-9695-1.
- Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 233–242. PMLR, 06–11 Aug 2017. URL <https://proceedings.mlr.press/v70/arpit17a.html>.
- Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=1NvflqAdoom>.
- Yuri Bakhtin. Noisy heteroclinic networks. *Probability theory and related fields*, 150(1):1–42, 2011.
- Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989. ISSN 0893-6080. doi: [https://doi.org/10.1016/0893-6080\(89\)90014-2](https://doi.org/10.1016/0893-6080(89)90014-2). URL <https://www.sciencedirect.com/science/article/pii/0893608089900142>.
- Ioannis Bantzi, James B. Simon, and Arthur Jacot. Saddle-to-saddle dynamics in deep relu networks: Low-rank bias in the first saddle escape, 2025. URL <https://arxiv.org/abs/2505.21722>.
- Raphaël Berthier. Incremental learning in diagonal linear networks. *Journal of Machine Learning Research*, 24(171):1–26, 2023. URL <http://jmlr.org/papers/v24/22-1395.html>.

Raphaël Berthier, Andrea Montanari, and Kangjie Zhou. Learning time-scales in two-layers neural networks. *Foundations of Computational Mathematics*, pp. 1–84, 2024.

Satwik Bhattacharya, Arkil Patel, Varun Kanade, and Phil Blunsom. Simplicity bias in transformers and their ability to learn sparse Boolean functions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5767–5791, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.317. URL <https://aclanthology.org/2023.acl-long.317/>.

Enric Boix-Adsera, Eta Littwin, Emmanuel Abbe, Samy Bengio, and Joshua Susskind. Transformers learn through gradual rank increase. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 24519–24551. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4d69c1c057a8bd570ba4a7b71aae8331-Paper-Conference.pdf.

Etienne Boursier and Nicolas Flammarion. Simplicity bias and optimization threshold in two-layer ReLU networks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 5241–5275. PMLR, 13–19 Jul 2025a. URL <https://proceedings.mlr.press/v267/boursier25a.html>.

Etienne Boursier and Nicolas Flammarion. Early alignment in two-layer networks training is a two-edged sword. *Journal of Machine Learning Research*, 26(183):1–75, 2025b. URL <http://jmlr.org/papers/v26/24-1523.html>.

Etienne Boursier, Lucas PILLAUD-VIVIEN, and Nicolas Flammarion. Gradient flow dynamics of shallow relu networks for square loss and orthogonal inputs. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 20105–20118. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/7eeb9af3eb1f48e29c05e8dd3342b286-Paper-Conference.pdf.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2205–2211. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/304. URL <https://doi.org/10.24963/ijcai.2021/304>. Main Track.

Ping-yeh Chiang, Renkun Ni, David Yu Miller, Arpit Bansal, Jonas Geiping, Micah Goldblum, and Tom Goldstein. Loss landscapes are all you need: Neural network generalization can be explained without the implicit bias of gradient descent. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=QC10RmRbZy9>.

Dmitry Chistikov, Matthias Englert, and Ranko Lazic. Learning a neuron by a shallow relu network: Dynamics and implicit bias for correlated inputs. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 23748–23760. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/4af24e6ce753c181e703f3f0be3b5e20-Paper-Conference.pdf.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/a614c557843b1df326cb29c57225459-Paper.pdf.

Kamaludin Dingle, Chico Q Camargo, and Ard A Louis. Input–output maps are strongly biased towards simple outputs. *Nature communications*, 9(1):761, 2018.

- Clémentine Carla Juliette Dominé, Nicolas Anguita, Alexandra Maria Proca, Lukas Braun, Daniel Kunin, Pedro A. M. Mediano, and Andrew M Saxe. From lazy to rich: Exact learning dynamics in deep linear networks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=ZXaoCMXc6d>.
- Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition, 2022. URL https://transformer-circuits.pub/2022/toy_model/index.html.
- Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=rJ1-b3RcF7>.
- Spencer Frei, Niladri S. Chatterji, and Peter L. Bartlett. Random feature amplification: Feature learning and generalization in neural networks. *Journal of Machine Learning Research*, 24(303):1–49, 2023a. URL <http://jmlr.org/papers/v24/22-1132.html>.
- Spencer Frei, Gal Vardi, Peter Bartlett, Nathan Srebro, and Wei Hu. Implicit bias in leaky reLU networks trained on high-dimensional data. In *The Eleventh International Conference on Learning Representations*, 2023b. URL <https://openreview.net/forum?id=JpbLyEI5EwW>.
- Kenji Fukumizu and Shun-ichi Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13(3):317–327, 2000. ISSN 0893-6080. doi: [https://doi.org/10.1016/S0893-6080\(00\)00009-5](https://doi.org/10.1016/S0893-6080(00)00009-5). URL <https://www.sciencedirect.com/science/article/pii/S0893608000000095>.
- Kenji Fukumizu, Shoichiro Yamaguchi, Yoh-ichi Mototake, and Mirai Tanaka. Semi-flat minima and saddle points by embedding neural networks to overparameterization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/a4ee59dd868ba016ed2de90d330acb6a-Paper.pdf.
- Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? a case study of simple function classes. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30583–30598. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c529dba08a146ea8d6cf715ae8930cbe-Paper-Conference.pdf.
- Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Bygh9j09KX>.
- Yoav Ger and Omri Barak. Learning dynamics of RNNs in closed-loop environments. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=P63bBMXCIH>.
- Borjan Geshkovski, Hugo Koubbi, Yury Polyanskiy, and Philippe Rigollet. Dynamic metastability in the self-attention model, 2024. URL <https://arxiv.org/abs/2410.06833>.
- Nikhil Ghosh, Song Mei, and Bin Yu. The three stages of learning dynamics in high-dimensional kernel methods. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=EQmAP4F859>.
- Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/f39ae9ff3a81f499230c4126e01f421b-Paper.pdf.

Daniel Gissin, Shai Shalev-Shwartz, and Amit Daniely. The implicit bias of depth: How incremental learning drives generalization. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=H1lj0nNfwB>.

Margalit Glasgow. SGD finds then tunes features in two-layer neural networks with near-optimal sample complexity: A case study in the XOR problem. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=HgOJlxzB16>.

Micah Goldblum, Marc Anton Finzi, Keefer Rowan, and Andrew Gordon Wilson. Position: The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15788–15808. PMLR, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/goldblum24a.html>.

Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Phys. Rev. X*, 10: 041044, Dec 2020. doi: 10.1103/PhysRevX.10.041044. URL <https://link.aps.org/doi/10.1103/PhysRevX.10.041044>.

Andrey Gromov, Kushal Tirumala, Hassan Shapourian, Paolo Glorioso, and Dan Roberts. The unreasonable ineffectiveness of the deeper layers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=ngmEcEer8a>.

Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9995–10006. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/71e9c6620d381d60196be694840aaaa-Paper.pdf.

G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. *Distributed representations*, pp. 77–109. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, pp. 5–13, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897916115. doi: 10.1145/168304.168306. URL <https://doi.org/10.1145/168304.168306>.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 01 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.1.1. URL <https://doi.org/10.1162/neco.1997.9.1.1>.

David Holzmüller and Ingo Steinwart. Training two-layer relu networks with gradient descent is inconsistent. *Journal of Machine Learning Research*, 23(181):1–82, 2022. URL <http://jmlr.org/papers/v23/20-830.html>.

Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17116–17128. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c6dfc6b7c601ac2978357b7a81e2d7ae-Paper.pdf.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=bCiNWDm1Y2>.

Francesco Innocenti, El Mehdi Achour, Ryan Singh, and Christopher L. Buckley. Only strict saddles in the energy landscape of predictive coding networks? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 53649–53683. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/6075fc6540b9a3cb951752099efd86ef-Paper-Conference.pdf.

Masato Inoue, Hyeyoung Park, and Masato Okada. On-line learning theory of soft committee machines with correlated hidden units –steepest gradient descent and natural gradient descent-. *Journal of the Physical Society of Japan*, 72(4):805–810, 2003. doi: 10.1143/JPSJ.72.805. URL <https://doi.org/10.1143/JPSJ.72.805>.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity. 2022. URL <https://arxiv.org/abs/2106.15933>.

Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=HJflg30qKX>.

Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 17176–17186. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c76e4b2fa54f8506719a5c0dc14c2eb9-Paper.pdf.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/b432f34c5a997c8e7c806a895ecc5e25-Paper.pdf.

Katie Kang, Amrith Setlur, Claire Tomlin, and Sergey Levine. Deep neural networks tend to extrapolate predictably. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=ljwoQ3cvQh>.

Kenji Kawaguchi. Deep learning without poor local minima. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/f2fc990265c712c49d51a18a32b39f0c-Paper.pdf.

Yiwen Kou, Zixiang Chen, and Quanquan Gu. Implicit bias of gradient descent for two-layer relu and leaky relu networks on nearly-orthogonal data. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 30167–30221. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/602f5c1b803c53b2aaf0b3864bf3383a-Paper-Conference.pdf.

Akshay Kumar and Jarvis Haupt. Directional convergence near small initializations and saddles in two-homogeneous neural networks. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=hfrPag75Y0>.

Daniel Kunin, Allan Raventós, Clémentine Dominé, Feng Chen, David Klindt, Andrew Saxe, and Surya Ganguli. Get rich quick: exact solutions reveal how unbalanced initializations promote rapid feature learning. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak,

and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 81157–81203. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/94074dd5a072d28ff75a76dabed43767-Paper-Conference.pdf.

Daniel Kunin, Giovanni Luca Marchetti, Feng Chen, Dhruva Karkada, James B Simon, Michael R DeWeese, Surya Ganguli, and Nina Miolane. Alternating gradient flows: A theory of feature learning in two-layer neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=t7LKc0MMW6>.

Thomas Laurent and James von Brecht. Deep linear networks with arbitrary loss: All local minima are global. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2902–2907. PMLR, 10–15 Jul 2018. URL <https://proceedings.mlr.press/v80/laurent18a.html>.

Thien Le and Stefanie Jegelka. Training invariances and the low-rank phenomenon: beyond linear networks. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=XEW8CQgArno>.

Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989. URL https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=AH0s7Sm5H7R>.

Yuhan Helena Liu, Aristide Baratin, Jonathan Cornford, Stefan Mihalas, Eric Todd SheaBrown, and Guillaume Lajoie. How connectivity structure shapes rich and lazy learning in neural circuits. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=s1SmYGc8ee>.

Haihao Lu and Kenji Kawaguchi. Depth creates no bad local minima, 2017. URL <https://arxiv.org/abs/1702.08580>.

Tao Luo, Zhi-Qin John Xu, Zheng Ma, and Yaoyu Zhang. Phase diagram for two-layer relu neural networks at infinite-width limit. *Journal of Machine Learning Research*, 22(71):1–47, 2021. URL <http://jmlr.org/papers/v22/20-1123.html>.

Kaifeng Lyu, Zhiyuan Li, Runzhe Wang, and Sanjeev Arora. Gradient descent on two-layer nets: Margin maximization and simplicity bias. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 12978–12991. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/6c351da15b5e8a743a21ee96a86e25df-Paper.pdf.

Hartmut Maennel, Olivier Bousquet, and Sylvain Gelly. Gradient descent quantizes relu network features, 2018. URL <https://arxiv.org/abs/1803.08367>.

Pierre Marion and Raphaël Berthier. Leveraging the two-timescale regime to demonstrate convergence of neural networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 64996–65029. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/cd062f8003e38f55dcb93df55b2683d6-Paper-Conference.pdf.

Hancheng Min, Enrique Mallada, and Rene Vidal. Early neuron alignment in two-layer reLU networks with small initialization. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=QibPzdVrRu>.

Chris Mingard, Guillermo Valle-Pérez, Joar Skalse, and Ard A. Louis. Is sgd a bayesian sampler? well, almost. *Journal of Machine Learning Research*, 22(79):1–64, 2021. URL <http://jmlr.org/papers/v22/20-676.html>.

Chris Mingard, Henry Rees, Guillermo Valle-Pérez, and Ard A Louis. Deep neural networks have an inbuilt occam’s razor. *Nature Communications*, 16(1):220, 2025. doi: <https://doi.org/10.1038/s41467-024-54813-x>. URL <https://doi.org/10.1038/s41467-024-54813-x>.

Philipp Misof, Pan Kessel, and Jan E Gerken. Equivariant neural tangent kernels. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 44470–44503. PMLR, 13–19 Jul 2025. URL <https://proceedings.mlr.press/v267/misof25a.html>.

Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=HkwBEMWCZ>.

Scott Pesme and Nicolas Flammarion. Saddle-to-saddle dynamics in diagonal linear networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 7475–7505. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/17a9ab4190289f0e1504bbb98d1d11a-Paper-Conference.pdf.

Leonardo Petrini, Francesco Cagnetta, Eric Vanden-Eijnden, and Matthieu Wyart. Learning sparse features can lead to overfitting in neural networks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 9403–9416. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/3d3a9e085540c65dd3e5731361f9320e-Paper-Conference.pdf.

Mary Phuong and Christoph H Lampert. The inductive bias of relu networks on orthogonally separable data. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=krz7T0xU9Z_.

Alexandra Maria Proca, Clémentine Carla Juliette Dominé, Murray Shanahan, and Pedro A. M. Mediano. Learning dynamics in linear recurrent neural networks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 49860–49902. PMLR, 13–19 Jul 2025. URL <https://proceedings.mlr.press/v267/proca25a.html>.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5301–5310. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/rahaman19a.html>.

Maria Refinetti, Alessandro Ingrosso, and Sebastian Goldt. Neural networks trained with SGD learn distributions of increasing complexity. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28843–28863. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/refinetti23a.html>.

Riccardo Rende, Federica Gerace, Alessandro Laio, and Sebastian Goldt. A distributional simplicity bias in the learning dynamics of transformers. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 96207–96228. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ae6c81a39079ddeb88b034b6ef18c7fe-Paper-Conference.pdf.

Jirko Rubruck, Jan Philipp Bauer, Andrew M Saxe, and Christopher Summerfield. Early learning of the optimal constant solution in neural networks and humans. In *8th Annual Conference on Cognitive Computational Neuroscience*, 2025. URL <https://openreview.net/forum?id=6Xyu486HRh>.

Roei Sarussi, Alon Brutkus, and Amir Globerson. Towards understanding learning in neural networks with linear teachers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9313–9322. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/sarussi21a.html>.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *The Second International Conference on Learning Representations*, 2014. URL https://openreview.net/forum?id=_wzZwKpTDF_9C.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019. doi: 10.1073/pnas.1820226116. URL <https://www.pnas.org/doi/abs/10.1073/pnas.1820226116>.

Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. Ray interference: a source of plateaus in deep reinforcement learning, 2019. URL <https://arxiv.org/abs/1904.11455>.

Jürgen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks*, 10(5):857–873, 1997. ISSN 0893-6080. doi: [https://doi.org/10.1016/S0893-6080\(96\)00127-X](https://doi.org/10.1016/S0893-6080(96)00127-X). URL <https://www.sciencedirect.com/science/article/pii/S089360809600127X>.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9573–9585. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6cfe0e6127fa25df2a0ef2ae1067d915-Paper.pdf.

Ohad Shamir. Exponential convergence time of gradient descent for one-dimensional deep linear neural networks. In Alina Beygelzimer and Daniel Hsu (eds.), *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pp. 2691–2713. PMLR, 25–28 Jun 2019. URL <https://proceedings.mlr.press/v99/shamir19a.html>.

James B Simon, Maksis Knutins, Liu Ziyin, Daniel Geisz, Abraham J Fetterman, and Joshua Albrecht. On the stepwise nature of self-supervised learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31852–31876. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/simon23a.html>.

Berfin Simsek, François Ged, Arthur Jacot, Francesco Spadaro, Clement Hongler, Wulfram Gerstner, and Johann Brea. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9722–9732. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/simsek21a.html>.

Salma Tarmoun, Guilherme Franca, Benjamin D Haeffele, and Rene Vidal. Understanding the dynamics of gradient flow in overparameterized linear models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10153–10161. PMLR, 18–24 Jul 2021. URL <https://proceedings.mlr.press/v139/tarmoun21a.html>.

Damien Teney, Ehsan Abbasnejad, Simon Lucey, and Anton van den Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16761–16772, June 2022.

Nadav Timor, Gal Vardi, and Ohad Shamir. Implicit regularization towards rank minimization in relu networks. In Shipra Agrawal and Francesco Orabona (eds.), *Proceedings of The 34th International Conference on Algorithmic Learning Theory*, volume 201 of *Proceedings of Machine Learning Research*, pp. 1429–1459. PMLR, 20 Feb–23 Feb 2023. URL <https://proceedings.mlr.press/v201/timor23a.html>.

Nikita Tsoy and Nikola Konstantinov. Simplicity bias of two-layer networks beyond linearly separable data. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 48728–48767. PMLR, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/tsoy24a.html>.

Zhenfeng Tu, Santiago Aranguri, and Arthur Jacot. Mixed dynamics in linear networks: Unifying the lazy and active regimes. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 106059–106104. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/bfa7d650ddfd7bd55866ee92f7a3393b-Paper-Conference.pdf.

Guillermo Valle-Perez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=rye4g3AqFm>.

Mingze Wang and Chao Ma. Understanding multi-phase optimization dynamics and rich nonlinear behaviors of reLU networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=konBXvt2iS>.

Haikun Wei, Jun Zhang, Florent Cousseau, Tomoko Ozeki, and Shun-ichi Amari. Dynamics of learning near singularities in layered networks. *Neural Computation*, 20(3):813–843, 03 2008. ISSN 0899-7667. doi: 10.1162/neco.2007.12-06-414. URL <https://doi.org/10.1162/neco.2007.12-06-414>.

Stephen Wiggins. *Invariant Manifolds: Linear and Nonlinear Systems*, pp. 28–70. Springer New York, New York, NY, 2003. ISBN 978-0-387-21749-9. doi: 10.1007/0-387-21749-5_4. URL https://doi.org/10.1007/0-387-21749-5_4.

Zhengqing Wu, Berfin Simsek, and François Gaston Ged. Loss landscape of shallow reLU-like neural networks: Stationary points, saddle escape, and network embedding. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=ogKE7LcvW6>.

Daniel Wurgafit, Ekdeep Singh Lubana, Core Francisco Park, Hidenori Tanaka, Gautam Reddy, and Noah Goodman. In-context learning strategies emerge rationally. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=bBUUQI0N6>.

Yu Yang, Eric Gan, Gintare Karolina Dziugaite, and Baharan Mirzasoleiman. Identifying spurious biases early in training through the lens of simplicity bias. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li (eds.), *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pp. 2953–2961. PMLR, 02–04 May 2024. URL <https://proceedings.mlr.press/v238/yang24c.html>.

Yuki Yoshida and Masato Okada. Data-dependence of plateau phenomenon in learning with neural network — statistical mechanical analysis. In H. Wallach, H. Larochelle, A. Beygelzimer,

- F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/287e03db1d99e0ec2edb90d079e142f3-Paper.pdf.
- Oğuz Kaan Yüksel, Rodrigo Alvarez Lucendo, and Nicolas Flammarion. Incremental learning of sparse attention patterns in transformers. In *EurIPS 2025 Workshop on Principles of Generative Modeling (PriGM)*, 2025. URL <https://openreview.net/forum?id=sZb2e9hM58>.
- Chulhee Yun, Suvrit Sra, and Ali Jadbabaie. Global optimality conditions for deep neural networks. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=BJk7Gf-CZ>.
- Yaoyu Zhang, Zhongwang Zhang, Tao Luo, and Zhiqin J Xu. Embedding principle of loss landscape of deep neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 14848–14859. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/7cc532d783a7461f227a5da8ea80bfe1-Paper.pdf.
- Yedi Zhang, Peter E. Latham, and Andrew M Saxe. Understanding unimodal bias in multimodal deep linear networks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 59100–59125. PMLR, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/zhang24aa.html>.
- Yedi Zhang, Andrew M Saxe, and Peter E. Latham. When are bias-free relu networks effectively linear networks? *Transactions on Machine Learning Research*, 2025a. ISSN 2835-8856. URL <https://openreview.net/forum?id=Ucpfdn66k2>.
- Yedi Zhang, Aaditya K Singh, Peter E. Latham, and Andrew M Saxe. Training dynamics of in-context learning in linear attention. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 76047–76087. PMLR, 13–19 Jul 2025b. URL <https://proceedings.mlr.press/v267/zhang25br.html>.
- Bo Zhao, Iordan Ganev, Robin Walters, Rose Yu, and Nima Dehmamy. Symmetries, flat minima, and the conserved quantities of gradient flow. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=9Zpc1CounFb>.
- Liu Ziyin, Yizhou Xu, Tomaso Poggio, and Isaac Chuang. Parameter symmetry potentially unifies deep learning theory. 2025. URL <https://arxiv.org/abs/2502.05300>.

Table of Contents

1	Introduction	1
2	Network Setup	3
3	Loss Landscape: Embedded Fixed Points	3
4	Invariant Manifold: Effectively Narrow Networks	5
5	Saddle-to-Saddle Dynamics	5
5.1	Linear case: timescale separation between directions	6
5.2	Quadratic case: timescale separation between units	7
6	Implications	8
7	Discussion	9
A	Additional Related Work	22
A.1	Saddle-to-saddle dynamics	22
A.2	Incremental learning in other settings	23
A.3	Simplicity bias	23
B	Additional Figures	24
B.1	Learning dynamics in two-layer networks	24
B.2	Learning dynamics in deep networks	26
B.3	The effect of skip connection	27
C	Additional Discussion	28
D	Gradient Calculations	29
E	Embedded Fixed Points	30
F	Invariant Manifolds	34
F.1	Definition of invariant manifolds	34
F.2	Proof of invariant manifolds	34
F.3	Expressivity on invariant manifolds	35
F.4	The embedded fixed points on invariant manifolds	36
G	Dynamics of Linear Networks	38
G.1	Gradient flow equations	38
G.2	Proof of timescale separation	38
G.3	Fixed points of linear networks	39
H	Dynamics of Quadratic Networks	41
H.1	Gradient flow equations	41
H.2	Derivations for timescale separation	41
I	Implementation Details	44

Appendix

A Additional Related Work

A.1 Saddle-to-saddle dynamics

Though saddle-to-saddle dynamics is a recurring phenomenon in the theory literature on learning dynamics, many questions remain open. The only case in which saddle-to-saddle dynamics has been proven for the full trajectory is the diagonal linear network in the limit of small initialization (Berthier, 2023; Pesme & Flammarion, 2023). For fully-connected linear networks, saddle-to-saddle dynamics has been shown under white input covariance and small spectral initialization (Saxe et al., 2014; 2019; Gidel et al., 2019; Gissin et al., 2020; Li et al., 2021). Outside this setting, the phase of visiting the first saddle from small initialization is well understood in linear networks (Jacot et al., 2022), while visiting subsequent saddles is not. For linear self-attention, Zhang et al. (2025b) showed saddle-to-saddle dynamics, but several phenomena remain unexplained, such as why it occurs even when the eigenvalues thought to govern the duration of plateaus are equal, and how increasing the number of heads affects the dynamics. While our work does not provide rigorous proofs for the technical open questions, we offer explanatory insights into these phenomena, and generate novel predictions on how network width, data statistics, and initialization affect saddle-to-saddle dynamics (Section 6).

There is also some work that did not focus on saddle-to-saddle dynamics but included relevant analysis. Zhang et al. (2024) showed that multimodal deep linear networks exhibit saddle-to-saddle dynamics. In multimodal deep linear networks with two input modalities, the saddle corresponds to a unimodal solution that is learning to fit the output only using one of the faster-to-learn modality. Rubruck et al. (2025) showed that two-layer linear networks with bias terms in the first layer exhibit saddle-to-saddle dynamics. Under some conditions, the first saddle corresponds to learning an optimal constant solution (Kang et al., 2024) that is the mean of the target output regardless of input.

Phenomena related to the timescale separation between directions in Theorem 4 have been examined in prior studies and are often discussed as weight alignment (Ji & Telgarsky, 2019; 2020; Atanasov et al., 2022). In a seminal work on the learning dynamics of deep linear networks (Saxe et al., 2014), aligned weights were introduced as an ansatz, with the analysis assuming aligned initial weights rather than deriving alignment from small isotropic initialization. This ansatz is often referred to in the linear network literature as the “spectral initialization” assumption; see Table 1 of Tarmoun et al. (2021) for a list of common initialization assumptions for linear networks. Later, Atanasov et al. (2022) analyzed the early phase dynamics of two-layer linear networks with scalar output from small isotropic initialization. They showed that the weights become increasingly aligned with a rank-one direction in the early phase, coined as “silent alignment”. Our Theorem 4 extends the “silent alignment” to the vector output case, and recovers it when the output is scalar. A useful note is that fully-connected linear networks with scalar output or scalar input do not have nonzero saddles, since a width-one network already has full expressivity. Consequently, they do not exhibit saddle-to-saddle dynamics except for escaping the saddle at zero.

Hu et al. (2020) showed that two-layer nonlinear networks from a particular symmetric initialization have similar dynamics to a linear model on the input in the early phase of training. Because the network outputs zero for any input at their symmetric initialization, their analysis is related to our analysis of linear network near small initialization (Section 5.1). Our Theorem 4 also leverages the fact that the network output is close to zero and the dynamics is approximately linear in the weights.

Kunin et al. (2025) proposed an algorithmic framework to capture staircase learning curves in two-layer networks. We go beyond the algorithmic level by presenting a theoretical framework that analyzes embedded fixed points, invariant manifolds, and two different timescale separation mechanisms. We complement their work by addressing the question of why gradient descent dynamics behaves similarly to their algorithm. Further, our results on fixed points and invariant manifolds are not limited to two-layer networks; they apply to deep networks defined by Equation (1). We also illuminate two distinct mechanisms for saddle-to-saddle dynamics depending on the architecture, i.e., the timescale separation between directions or units. These two mechanisms were not distinguished in prior literature.

Ziyin et al. (2025) proposed a general hypothesis that learning dynamics is symmetry-to-symmetry. The saddle-to-saddle dynamics in our work is related to permutation symmetry between the units. Our work makes a theoretical case for their hypothesis and clarifies the conditions under which saddle-to-saddle dynamics occurs.

A.2 Incremental learning in other settings

Incremental learning, characterized by earlier phases corresponding to simpler solutions, has been examined in several other theoretical settings. Cao et al. (2021); Ghosh et al. (2022) studied the spectral bias in the neural tangent kernel regime. In the kernel regime, eigenfunctions with larger eigenvalues are learned faster. This behavior differs from saddle-to-saddle dynamics, as the network in the kernel regime neither visits saddles nor exhibits plateaus during learning. Instead, the training loss decreases throughout learning, with faster decay early in learning and slower decay later. Outside the kernel regime, Abbe et al. (2023) studied a layer-wise training setup, in which the first and second layers are trained separately. Marion & Berthier (2023); Berthier et al. (2024) studied a two-timescale regime, in which the second-layer weights are trained with a much larger learning rate than the first-layer weights. In comparison, our analysis focuses on the learning dynamics of standard gradient descent outside the neural tangent kernel regime.

A.3 Simplicity bias

In this paper, we focus on the dynamical simplicity bias; that is, learning increasingly complex solutions over the course of training. A broader, longstanding body of theoretical and experimental research has explored the “stationary” simplicity bias, independent of training dynamics. Early studies (Hinton & van Camp, 1993; Hochreiter & Schmidhuber, 1997) connected generalization to the minimum description length of the weights, suggesting that flat minima correspond to simple solutions that potentially generalize well. A line of work based on algorithmic information theory (Schmidhuber, 1997; Valle-Perez et al., 2019; Dingle et al., 2018; Goldblum et al., 2024; Mingard et al., 2021; 2025) showed that standard architectures with randomly sampled weights are biased toward input-output maps with low Kolmogorov complexity. Empirical work (Huh et al., 2023) documented that both randomly initialized networks and trained networks exhibit a bias toward input-output maps with low effective rank embeddings. These findings motivated a volume hypothesis: neural networks have a simplicity bias arising from the loss landscape; specifically, the simple solutions occupy a larger volume in the weight space than the complex ones (Huh et al., 2023; Chiang et al., 2023). Our work on dynamical simplicity bias complements the stationary simplicity bias literature by examining how gradient descent dynamics drives the progression of solution complexity over time.

According to the no free lunch theorem, no single inductive bias is universally beneficial. Thus, the simplicity bias can be either advantageous or detrimental depending on the task. Several studies (Hermann & Lampinen, 2020; Shah et al., 2020; Petruini et al., 2022; Yang et al., 2024) have shown that favoring simple solutions may harm generalization when the simple solution relies on simple but spurious features, whereas more complex but robust features yield better generalization. For example, convolutional neural networks often prefer to classify objects by texture rather than shape, even though the classifier relying on shape can generalize better (Geirhos et al., 2019). Moreover, studies on two-layer ReLU networks (Holzmüller & Steinwart, 2022; Tsoy & Konstantinov, 2024; Boursier & Flammarion, 2025a;b) demonstrated that simplicity bias can sometimes cause optimization difficulties, where the first-layer weights align with a limited set of spurious directions when omnidirectional weights are required to reach global minima.

B Additional Figures

B.1 Learning dynamics in two-layer networks

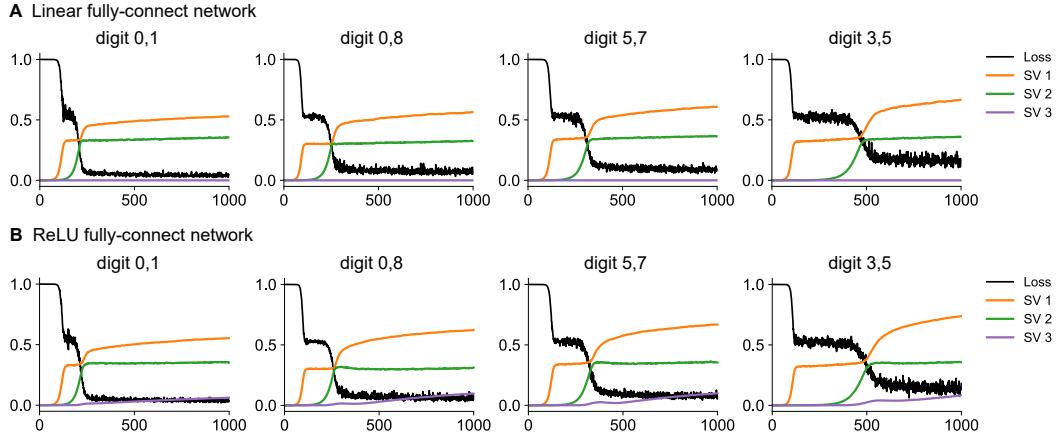


Figure 3: Saddle-to-saddle dynamics in two-layer fully-connected linear and ReLU networks trained for binary classification of MNIST digits. The input dimension is $28 \times 28 = 784$, the hidden layer width is 1000, and the target outputs are two-dimensional one-hot vectors. The intermediate plateau is longer when the two digits are harder to distinguish. For example, digits 3/5 are harder to distinguish than digits 0/1. The colored curves represent the top three singular values of the first-layer weight matrix, $\bar{U} \in \mathbb{R}^{1000 \times 784}$. Consistent with our theory, the growth of the first and second singular values coincides with the first and second abrupt drops in the training loss, respectively, corresponding to the increase in the effective width. The third largest singular value is close to zero, meaning the rank of the first-layer weight matrix is at most two, approximately. Details: The batch size is 64. The learning rate is 0.01. The initial weights are sampled independently from $\mathcal{N}(0, 10^{-12})$.

Table 1: Singular values of MNIST binary classification data.

Digit pair	0,1	0,8	5,7	3,5
Singular value governing the first plateau	4.20	5.21	4.13	4.62
Singular value governing the second plateau	3.90	5.21	3.26	1.38

In Figure 3, we show the learning dynamics of two-layer linear and ReLU networks trained for binary classification of MNIST digits. Despite being noisier than the learning dynamics on synthetic datasets, the plateaus and abrupt drops in the training loss are still pronounced. The abrupt drop in loss also coincides with the growth of a singular value of the first-layer weight matrix, implying that an increase in the effective width can explain the observed dynamics. We show the singular values governing the duration of the first and second plateaus of the linear network dynamics in Table 1, which are the first singular values of Σ_{yx} and $(I - e_1 e_1^\top) \Sigma_{yx}$, respectively. Here e_1 is the first eigenvector of the matrix $\Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{yx}^\top$; see Theorem 6. The sizes of the singular values approximately match the duration of the plateaus in Figure 3.

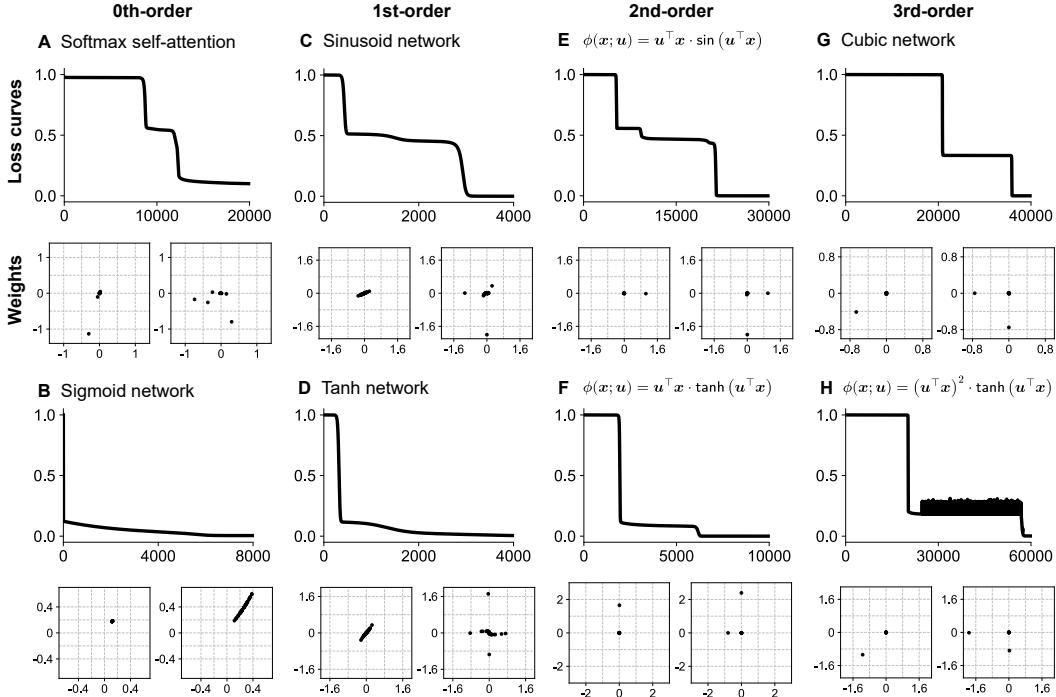


Figure 4: Learning dynamics in two-layer networks with other activation functions. Each panel shows the loss during training (top), and the first-layer weights right after the first abrupt loss drop (bottom left) and at the end of learning (bottom right). The first-layer weights to each hidden unit are two-dimensional and plotted as black dots. (A) The softmax self-attention model is the same as linear self-attention in Figure 1F, except for adding the softmax activation function. The training data set is the same as that of Figure 1F. (B) Two-layer fully-connected sigmoid network, i.e., $\phi(\mathbf{x}; \mathbf{u}) = \text{sigmoid}(\mathbf{u}^\top \mathbf{x})$. (C) Two-layer fully-connected sinusoid network, i.e., $\phi(\mathbf{x}; \mathbf{u}) = \sin(\mathbf{u}^\top \mathbf{x})$. (D) Two-layer fully-connected tanh network, i.e., $\phi(\mathbf{x}; \mathbf{u}) = \tanh(\mathbf{u}^\top \mathbf{x})$. (E,F,H) Two-layer fully-connected networks with the given activation functions. (G) Two-layer fully-connected cubic network, i.e., $\phi(\mathbf{x}; \mathbf{u}) = (\mathbf{u}^\top \mathbf{x})^3$. Details: Except for panel A, the training set is generated by a width-2 teacher network with the same activation function, $y = \phi(\mathbf{x}; \mathbf{u}_1^*) + \phi(\mathbf{x}; \mathbf{u}_2^*)$, $\mathbf{x} \in \mathbb{R}^2$, $y \in \mathbb{R}$. The input is sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. For panels B-F, the teacher network has $\mathbf{u}_1^* = [1, 0]^\top$, $\mathbf{u}_2^* = [0, 2]^\top$. For panels G,H, the teacher network has $\mathbf{u}_1^* = [1, 0]^\top$, $\mathbf{u}_2^* = [0, 1]^\top$. The number of training samples is 8192. The learning rate is 0.02. The initial weights are sampled independently from $\mathcal{N}(0, 10^{-12})$ for panels A-D, and from $\mathcal{N}(0, 0.005^2)$ for panels E-H. The width is $H = 50$ for panels A-D, and $H = 10$ for panels E-H.

In Figure 4, we plot the loss and weights dynamics for two-layer networks with other activation functions. If we Taylor expand $\phi(\mathbf{x}; \mathbf{u})$ around $\mathbf{u} = \mathbf{0}$, the lowest-order non-vanishing terms are the 0th-order, 1st-order, 2nd-order, and 3rd-order terms for panels (A,B), (C,D), (E,F), and (G,H), respectively. In panels (C,D), the networks develop rank-one weights in the early dynamics, since the sinusoid and tanh activation functions are approximately linear around zero. After the first abrupt drop in loss, our theory cannot predict the dynamics anymore because rank-one weights do not generally correspond to embedded fixed points or invariant manifolds for sinusoid and tanh networks. In panels (E,F,G,H), the networks undergo saddle-to-saddle dynamics, similar to the quadratic networks studied in Section 5.2. This is because dynamics with the 2nd-order and 3rd-order terms exhibit a timescale separation between units, as discussed at the end of Section 5. Indeed, the weights dynamics in panels (E,F,G,H) show that there are one and two units with large weights with the rest being near zero during the intermediate plateau and at convergence, respectively.

B.2 Learning dynamics in deep networks

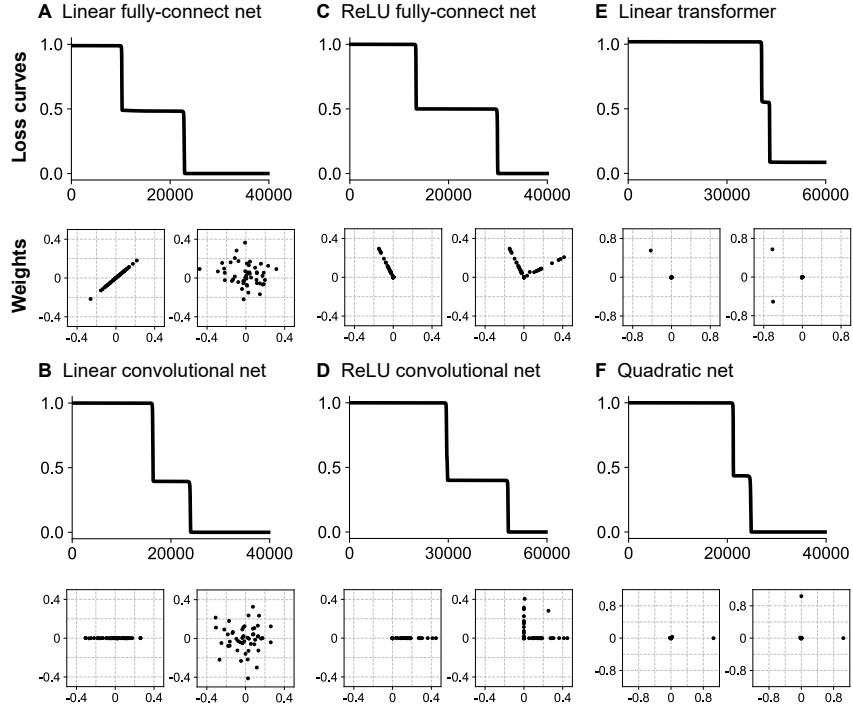


Figure 5: Learning dynamics in deep networks. Each panel shows the loss over training time (top), and the first-layer weights right after the first abrupt loss drop (bottom left) and at the end of learning (bottom right). The first-layer weights to each hidden unit are two-dimensional and plotted as black dots. The training sets in panels A,C,E are the same as those in Figure 1(B,D,F). The training sets in panels B,D,F split the scalar output in Figure 1(C,E,G) into a two-dimensional vector output. (A) Three-layer linear fully-connected network. (B) The network has a convolutional linear layer as the first hidden layer and a fully-connected linear layer as the second hidden layer. (C) Three-layer ReLU fully-connected network. (D) The network has a convolutional ReLU layer as the first hidden layer and a fully-connected ReLU layer as the second hidden layer. (E) One-layer linear transformer, consisting of one linear self-attention layer and two fully-connected linear layers. (F) The network has a fully-connected layer with quadratic activation as the first hidden layer and a fully-connected linear layer as the second hidden layer. Details: The number of training samples is 8192. The learning rate is 0.02. The initial weights are sampled independently from $\mathcal{N}(0, 0.005^2)$ for panels A-E, and from $\mathcal{N}(0, 0.05^2)$ for panel F. The width is $H = 50$ for panels A-D, and $H = 10$ for panels E,F.

In Figure 5, we present the learning dynamics of deep networks with various architectures, in comparison with the two-layer architectures in Figure 1. Similar to two-layer networks, the deep networks also exhibit saddle-to-saddle dynamics. The weight structures indicate that the visited saddles in panels (A,B) correspond to the embedded fixed points in Equation (7), those in panels (C,D) correspond to Equation (6), and those in panels (E,F) correspond to Equation (5). The effective width of the first layer during the intermediate plateau and at convergence is one and two, respectively.

We let the output of the networks in Figure 5(B,D,F) be two-dimensional, rather than one-dimensional as in Figure 1(C,E,G), due to considerations of expressivity. If the output is one-dimensional, the second fully-connected linear layer can achieve full expressivity with an effective width of one. This would make the second saddle-to-saddle transition (if there is one) different from the first: in the first transition, the effective width of both layers increases by one, whereas in the second transition only the effective width of the first layer increases. We leave this interesting problem to future research.

B.3 The effect of skip connection

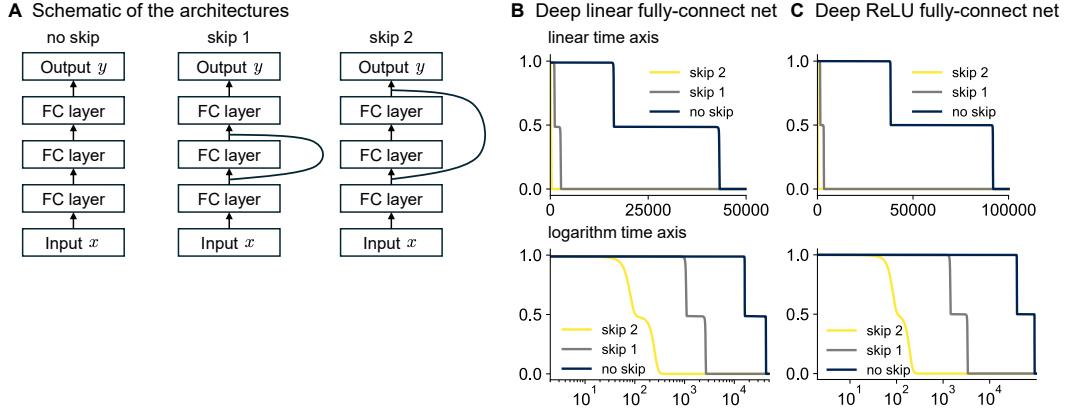


Figure 6: Saddle-to-saddle dynamics in deep fully-connected networks with skip connections. (A) Schematic of three four-layer fully-connected networks: one with no skip connection, one with a skip connection that skips one layer, and one with a skip connection that skips two layers. The three linear networks are defined in Equation (18). (B,C) Loss curves of linear and ReLU networks with skip connections, plotted using linear time (top row) and logarithmic time (bottom row) axes. All networks exhibit saddle-to-saddle dynamics, with the network that skips more layers learning faster. With small initialization, shallower linear networks learn faster (Saxe et al., 2019). In the network without a skip connection, all four layers must escape the zero fixed point and learn. In the network that skips one layer, the second-layer weights can remain near zero while the other three layers escape the zero fixed point and learn, yielding dynamics similar to a three-layer network. In the network that skips two layers, only the first and last layers need to learn, yielding dynamics similar to a two-layer network.

In Figure 6, the networks are four-layer linear networks with skip connections, defined as

$$\text{no skip: } f(\mathbf{x}) = \mathbf{W}_4 \mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1 \mathbf{x} \quad (18a)$$

$$\text{skip 1: } f(\mathbf{x}) = \mathbf{W}_4 \mathbf{W}_3 (\mathbf{W}_2 \mathbf{W}_1 \mathbf{x} + \mathbf{W}_1 \mathbf{x}) \quad (18b)$$

$$\text{skip 2: } f(\mathbf{x}) = \mathbf{W}_4 (\mathbf{W}_3 \mathbf{W}_2 \mathbf{W}_1 \mathbf{x} + \mathbf{W}_1 \mathbf{x}) \quad (18c)$$

where the input $\mathbf{x} \in \mathbb{R}^2$ and the weights $\mathbf{W}_4 \in \mathbb{R}^{2 \times 50}$, $\mathbf{W}_3, \mathbf{W}_2 \in \mathbb{R}^{50 \times 50}$, $\mathbf{W}_1 \in \mathbb{R}^{50 \times 2}$.

All three networks defined in Equation (18) exhibit saddle-to-saddle dynamics when trained from small initialization, with the network that skips more layers learning faster. This is because weights in the skipped layers can remain near zero while weights in other layers escape the zero fixed point and learn. When the skipped layers are effectively unused, the network behaves like a shallower network consisting only of the unskipped layers and exhibits saddle-to-saddle dynamics. Furthermore, since shallower linear networks learn faster (Saxe et al., 2019), networks that skip more layers also learn faster. The dynamics in ReLU networks with skip connections is similar.

C Additional Discussion

ReLU activation function. Because the ReLU activation function is piece-wise linear, we conjecture that ReLU networks trained from small initialization have a timescale separation between different directions, similar to the mechanism in linear networks. Indeed, prior studies have found that the direction that maximizes the correlation with the output grows the fastest, that is the direction $\arg \max_{\|u\|=1} \langle \text{ReLU}(u^\top x) y \rangle$. This was known as quantizing (Maennel et al., 2018), condensing (Luo et al., 2021), random feature amplification (Frei et al., 2023a), and studied in many other theoretical works on the learning dynamics of ReLU networks (Le & Jegelka, 2022; Petrini et al., 2022; Timor et al., 2023; Kou et al., 2023; Glasgow, 2024; Min et al., 2024).

Reduction of high-dimensional learning dynamics. The invariant manifolds in Theorem 3 can also be useful for reducing high-dimensional learning dynamics. Whenever a network evolves near an invariant manifold during a learning phase, Theorem 3 suggests that the full learning dynamics may be well approximated by the dynamics of a narrower network. For example, if a width-2 homogeneous network has nearly proportional weights, $\theta_1 \approx \gamma \theta_2$, dynamics in θ_1 and θ_2 may be well approximated by lower-dimensional dynamics only in θ_1 . Many prior analyses have successfully reduced high-dimensional learning dynamics for individual architectures using a similar idea, including deep linear networks (Saxe et al., 2014; 2019; Advani et al., 2020), ReLU networks (Phuong & Lampert, 2021; Sarussi et al., 2021; Lyu et al., 2021; Frei et al., 2023b; Tsoy & Konstantinov, 2024; Zhang et al., 2025a), and self-attention (Zhang et al., 2025b; Yüksel et al., 2025).

Distributed or localized features. Identifying which type of fixed points a network visits among Equations (4) to (7) may be of interest to representation learning (Hinton et al., 1986; Elhage et al., 2022) and pruning problems (LeCun et al., 1989; Frankle & Carbin, 2019; Gromov et al., 2025). The fixed points described by Equations (4), (6) and (7) correspond to networks with distributed, nonlocal, or polysemantic features, while fixed points described by Equation (5) correspond to networks with localized or monosemantic features. Networks with localized features and simplicity bias can be more easily pruned. In Section 5, we showed that the timescale separation between directions due to data distribution gives rise to distributed features, while the timescale separation between units gives rise to localized features.

Technical future directions. The goal of this paper is to establish a theoretical framework and validate its predictive power. To serve this goal, we have prioritized the completeness of the framework, at times relying on heuristics and empirical observations. Several interesting technical questions remain open. First, how close must a point in weight space be to an invariant manifold in order to approach a fixed point on that manifold before leaving the manifold? Quantifying this could enable rigorous proofs of saddle-to-saddle dynamics in a wider range of architectures, extending beyond diagonal linear networks (Berthier, 2023; Pesme & Flammariou, 2023). Second, is the sequence of saddles visited during training Markovian? That is, can the next saddle be inferred solely from the current one, independent of earlier ones? In dynamical systems literature, it has been shown that certain saddle-to-saddle transitions are non-Markovian (Bakhtin, 2011).

D Gradient Calculations

We write down the gradients of the weights $\theta_{1:H}$ for the network defined in Equation (1). We denote

$$f(\mathbf{x}) = g_{\text{out}}(\boldsymbol{\zeta}), \quad \text{where } \boldsymbol{\zeta} = \sum_{i=1}^H \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i) \mathbf{v}_i. \quad (19)$$

The variables have dimensionality

$$\mathbf{u} \in \mathbb{R}^{N_u}, \mathbf{v} \in \mathbb{R}^{N_v}, \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}) \in \mathbb{R}^{N_\phi \times N_v}, \boldsymbol{\zeta} \in \mathbb{R}^{N_\phi}.$$

Recall that the training loss is defined as

$$\mathcal{L} = \frac{1}{P} \sum_{\mu=1}^P \ell_\mu = \frac{1}{P} \sum_{\mu=1}^P \ell(\mathbf{y}_\mu, f(\mathbf{x}_\mu)). \quad (20)$$

Using the chain rule, the gradient flow dynamics of \mathbf{v}_i can be written

$$\dot{\mathbf{v}}_i = -\frac{\partial \mathcal{L}}{\partial \mathbf{v}_i} = -\frac{1}{P} \sum_{\mu=1}^P \left(\frac{\partial \boldsymbol{\zeta}}{\partial \mathbf{v}_i} \right)^\top \frac{\partial \ell_\mu}{\partial \boldsymbol{\zeta}} = -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^\top \frac{\partial \ell_\mu}{\partial \boldsymbol{\zeta}}. \quad (21)$$

The gradient flow dynamics of \mathbf{u}_i involves matrix-by-vector derivatives, which generally require tensor notations. To avoid introducing tensor notations, we instead write the gradient entrywise, for $n = 1, \dots, N_u$,

$$\dot{u}_{i,n} = -\frac{\partial \mathcal{L}}{\partial u_{i,n}} = -\frac{1}{P} \sum_{\mu=1}^P \left(\frac{\partial \boldsymbol{\zeta}}{\partial u_{i,n}} \right)^\top \frac{\partial \ell_\mu}{\partial \boldsymbol{\zeta}} = -\frac{1}{P} \sum_{\mu=1}^P \mathbf{v}_i^\top \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^\top}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \boldsymbol{\zeta}}. \quad (22)$$

E Embedded Fixed Points

Here we prove Theorem 1.

Proof. We have that $\boldsymbol{\theta}_{1:(H-1)}^*$ is a fixed point of the gradient flow dynamics of the width- $(H-1)$ network, that is

$$\dot{\boldsymbol{\theta}}_i = -\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_i} \Big|_{\boldsymbol{\theta}_i=\boldsymbol{\theta}_i^*} = -\frac{\partial \mathcal{L}}{\partial f^*(\mathbf{x})} \frac{\partial f^*(\mathbf{x})}{\partial \boldsymbol{\theta}_i^*} = \mathbf{0}, \quad i = 1, \dots, H-1. \quad (23)$$

We denote

$$\boldsymbol{\zeta}^* = \sum_{j=1}^{H-1} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j^*) \mathbf{v}_j^*. \quad (24)$$

The input-output map the width- $(H-1)$ network is $f^*(\mathbf{x}) = g_{\text{out}}(\boldsymbol{\zeta}^*)$.

We prove the four statements one by one.

(i) For any ϕ , we analyze Equation (4).

First, the width- H network implements the same input-output map as the width- $(H-1)$ network, that is $f(\mathbf{x}) = g_{\text{out}}(\boldsymbol{\zeta})$ where

$$\begin{aligned} \boldsymbol{\zeta} &= \sum_{j=1}^H \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j) \mathbf{v}_j \\ &= \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_H) \mathbf{v}_H + \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i) \mathbf{v}_i + \sum_{j=1, j \neq i}^{H-1} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j) \mathbf{v}_j \\ &= \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i^*) \gamma_v \mathbf{v}_i^* + \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i^*) (1 - \gamma_v) \mathbf{v}_i^* + \sum_{j=1, j \neq i}^{H-1} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j^*) \mathbf{v}_j^* \\ &= \sum_{j=1}^{H-1} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j^*) \mathbf{v}_j^* \\ &= \boldsymbol{\zeta}^*. \end{aligned}$$

Second, we calculate the gradients of the weights in the width- H network. For the units with unmodified weights, the gradients are the same as the width- $(H-1)$ network

$$\dot{\boldsymbol{\theta}}_j = -\frac{\partial \mathcal{L}}{\partial f(\mathbf{x})} \frac{\partial f(\mathbf{x})}{\partial \boldsymbol{\theta}_j} = -\frac{\partial \mathcal{L}}{\partial f^*(\mathbf{x})} \frac{\partial f^*(\mathbf{x})}{\partial \boldsymbol{\theta}_j^*} = \mathbf{0}, \quad j = 1, \dots, H-1, j \neq i.$$

For the i -th unit, the gradients can be expressed using Equations (21) and (22) as

$$\begin{aligned} \dot{\mathbf{v}}_i &= -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^\top \frac{\partial \ell_\mu}{\partial \boldsymbol{\zeta}} = -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i^*)^\top \frac{\partial \ell_\mu}{\partial \boldsymbol{\zeta}^*} = \dot{\mathbf{v}}_i^* = \mathbf{0}, \\ \dot{u}_{i,n} &= -\frac{1}{P} \sum_{\mu=1}^P \mathbf{v}_i^\top \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^\top}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \boldsymbol{\zeta}} \\ &= -(1 - \gamma_v) \frac{1}{P} \sum_{\mu=1}^P \mathbf{v}_i^{*\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i^*)^\top}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \boldsymbol{\zeta}^*} \\ &= (1 - \gamma_v) \dot{u}_{i,n}^* \\ &= 0, \quad n = 1, \dots, N_u. \end{aligned}$$

Similarly, for the new H -th unit, the gradients are

$$\begin{aligned}\dot{\boldsymbol{v}}_H &= -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\boldsymbol{x}_\mu); \boldsymbol{u}_i^*)^\top \frac{\partial \ell_\mu}{\partial \zeta^*} = \dot{\boldsymbol{v}}_i^* = \mathbf{0}, \\ \dot{u}_{H,n} &= -\gamma_v \frac{1}{P} \sum_{\mu=1}^P \boldsymbol{v}_i^{*\top} \frac{\partial \phi(g_{\text{in}}(\boldsymbol{x}_\mu); \boldsymbol{u}_i^*)^\top}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \zeta^*} = \gamma_v \dot{u}_{i,n}^* = 0, \quad n = 1, \dots, N_u.\end{aligned}$$

(ii) For ϕ such that $\forall z, \phi(z; \mathbf{u}_{\text{zero}}) = 0$, we analyze Equation (5).

The width- H network implements the same input-output map as the width- $(H-1)$ network, that is $f(\boldsymbol{x}) = g_{\text{out}}(\zeta)$ where

$$\zeta = \sum_{i=1}^H \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_i) \boldsymbol{v}_i = \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_{\text{zero}}) \mathbf{0} + \sum_{j=1}^{H-1} \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_j^*) \boldsymbol{v}_j^* = \zeta^*.$$

Because the first $(H-1)$ units have unmodified weights, their gradients are the same as those in the width- $(H-1)$ network, which are zero. For the new H -th unit, the gradients can be expressed using Equations (21) and (22) as

$$\begin{aligned}\dot{\boldsymbol{v}}_H &= -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\boldsymbol{x}_\mu); \boldsymbol{u}_{\text{zero}})^\top \frac{\partial \ell_\mu}{\partial \zeta} = \mathbf{0}, \\ \dot{u}_{H,n} &= -\frac{1}{P} \sum_{\mu=1}^P \mathbf{0}^\top \frac{\partial \phi(g_{\text{in}}(\boldsymbol{x}_\mu); \boldsymbol{u}_H)^\top}{\partial u_{H,n}} \frac{\partial \ell_\mu}{\partial \zeta} = 0, \quad n = 1, \dots, N_u.\end{aligned}$$

(iii) If $\phi(z; \mathbf{u})$ is degree-1 homogeneous in \mathbf{u} , we analyze Equation (6).

The width- H network implements the same input-output map as the width- $(H-1)$ network, that is $f(\boldsymbol{x}) = g_{\text{out}}(\zeta)$ where

$$\begin{aligned}\zeta &= \sum_{j=1}^H \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_j) \boldsymbol{v}_j \\ &= \phi(g_{\text{in}}(\boldsymbol{x}); \gamma_u \boldsymbol{u}_i^*) \gamma_v \boldsymbol{v}_i^* + \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_i^*) (1 - \gamma_u \gamma_v) \boldsymbol{v}_i^* + \sum_{j=1, j \neq i}^{H-1} \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_j^*) \boldsymbol{v}_j^* \\ &= \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_i^*) \gamma_u \gamma_v \boldsymbol{v}_i^* + \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_i^*) (1 - \gamma_u \gamma_v) \boldsymbol{v}_i^* + \sum_{j=1, j \neq i}^{H-1} \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_j^*) \boldsymbol{v}_j^* \\ &= \sum_{j=1}^{H-1} \phi(g_{\text{in}}(\boldsymbol{x}); \boldsymbol{u}_j^*) \boldsymbol{v}_j^* \\ &= \zeta^*.\end{aligned}$$

Because the units with indices $j = 1, \dots, H-1, j \neq i$ have unmodified weights, their gradients are the same as those in the width- $(H-1)$ network, which are zero. For the i -th

unit, the gradients can be expressed using Equations (21) and (22) as

$$\begin{aligned}
\dot{\mathbf{v}}_i &= -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i^*)^\top \frac{\partial \ell_\mu}{\partial \zeta^*} = \dot{\mathbf{v}}_i^* = \mathbf{0}, \\
\dot{\mathbf{u}}_{i,n} &= -\frac{1}{P} \sum_{\mu=1}^P \mathbf{v}_i^\top \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \zeta} \\
&= -\frac{1}{P} \sum_{\mu=1}^P (1 - \gamma_u \gamma_v) \mathbf{v}_i^{*\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i^*)^\top}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \zeta^*} \\
&= (1 - \gamma_u \gamma_v) \dot{\mathbf{u}}_{i,n}^* \\
&= 0, \quad n = 1, \dots, N_u.
\end{aligned}$$

By Euler's homogeneous function theorem, the partial derivative of the homogeneous function $\phi(\mathbf{z}; \mathbf{u})$ has the following property

$$\left. \frac{\partial \phi(\mathbf{z}; \mathbf{u})}{\partial u_n} \right|_{\mathbf{u}=\gamma \mathbf{u}^*} = \left. \frac{\partial \phi(\mathbf{z}; \mathbf{u})}{\partial u_n} \right|_{\mathbf{u}=\mathbf{u}^*}, \quad n = 1, \dots, N_u. \quad (25)$$

Thus, for the new H -th unit, the gradients are

$$\begin{aligned}
\dot{\mathbf{v}}_H &= -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_H)^\top \frac{\partial \ell_\mu}{\partial \zeta} = -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \gamma_u \mathbf{u}_i^*)^\top \frac{\partial \ell_\mu}{\partial \zeta^*} = \gamma_u \dot{\mathbf{v}}_i^* = \mathbf{0}, \\
\dot{\mathbf{u}}_{H,n} &= -\frac{1}{P} \sum_{\mu=1}^P \mathbf{v}_H^\top \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_H)}{\partial u_{H,n}} \frac{\partial \ell_\mu}{\partial \zeta^*} \\
&= -\frac{1}{P} \sum_{\mu=1}^P \gamma_v \mathbf{v}_i^{*\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \gamma_u \mathbf{u}_i^*)^\top}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \zeta^*} \\
&= \gamma_v \dot{\mathbf{u}}_{i,n}^* = 0, \quad n = 1, \dots, N_u.
\end{aligned}$$

(iv) If $\phi(\mathbf{z}; \mathbf{u})$ is linear in \mathbf{u} , we analyze Equation (7).

The width- H network implements the same input-output map as the width- $(H-1)$ network, that is $f(\mathbf{x}) = g_{\text{out}}(\zeta)$ where

$$\begin{aligned}
\zeta &= \sum_{j=1}^H \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j) \mathbf{v}_j \\
&= \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_H) \mathbf{v}_H + \sum_{j=1}^{H-1} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j) \mathbf{v}_j \\
&= \phi \left(g_{\text{in}}(\mathbf{x}); \sum_{i=1}^{H-1} \gamma_{u_i} \mathbf{u}_i^* \right) \left(\sum_{i=1}^{H-1} \gamma_{v_i} \mathbf{v}_i^* \right) + \sum_{j=1}^{H-1} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j^*) \left(\mathbf{v}_j^* - \gamma_{u_j} \sum_{j'=1}^{H-1} \gamma_{v_{j'}} \mathbf{v}_{j'}^* \right) \\
&= \sum_{i,i'=1}^{H-1} \gamma_{u_i} \gamma_{v_{i'}} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_i^*) \mathbf{v}_{i'}^* + \sum_{j=1}^{H-1} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j^*) \mathbf{v}_j^* - \sum_{j,j'=1}^{H-1} \gamma_{u_j} \gamma_{v_{j'}} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j^*) \mathbf{v}_{j'}^* \\
&= \sum_{j=1}^{H-1} \phi(g_{\text{in}}(\mathbf{x}); \mathbf{u}_j^*) \mathbf{v}_j^* \\
&= \zeta^*.
\end{aligned}$$

For $i = 1, \dots, H-1$, the gradients can be expressed using Equations (21) and (22) as

$$\begin{aligned}
\dot{\mathbf{v}}_i &= -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i^*)^\top \frac{\partial \ell_\mu}{\partial \zeta^*} = \dot{\mathbf{v}}_i^* = \mathbf{0}, \\
\dot{u}_{i,n} &= -\frac{1}{P} \sum_{\mu=1}^P \left(\mathbf{v}_i^* - \gamma_{u_i} \sum_{j=1}^{H-1} \gamma_{v_j} \mathbf{v}_j^* \right)^\top \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^\top}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \zeta} \\
&= -\frac{1}{P} \sum_{\mu=1}^P \mathbf{v}_i^{*\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^\top}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \zeta} - \gamma_{u_i} \sum_{j=1}^{H-1} \gamma_{v_j} \frac{1}{P} \sum_{\mu=1}^P \mathbf{v}_j^{*\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_j)^\top}{\partial u_{j,n}} \frac{\partial \ell_\mu}{\partial \zeta} \\
&= \dot{u}_{i,n}^* + \gamma_{u_i} \sum_{j=1}^{H-1} \gamma_{v_j} \dot{u}_{j,n}^* \\
&= 0, \quad n = 1, \dots, N_u.
\end{aligned}$$

We leverage the degree-1 homogeneous and additive properties of linearity, which yields

$$\phi \left(\mathbf{z}; \sum_{i=1}^{H-1} \gamma_{u_i} \mathbf{u}_i^* \right) = \sum_{i=1}^{H-1} \gamma_{u_i} \phi(\mathbf{z}; \mathbf{u}_i^*) \quad (26)$$

Thus, for the new H -th unit, the gradient for \mathbf{v}_H is

$$\begin{aligned}
\mathbf{v}_H &= -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_H)^\top \frac{\partial \ell_\mu}{\partial \zeta} \\
&= -\frac{1}{P} \sum_{\mu=1}^P \phi \left(g_{\text{in}}(\mathbf{x}_\mu); \sum_{i=1}^{H-1} \gamma_{u_i} \mathbf{u}_i^* \right)^\top \frac{\partial \ell_\mu}{\partial \zeta^*} \\
&= \sum_{i=1}^{H-1} \gamma_{u_i} \left(-\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i^*)^\top \frac{\partial \ell_\mu}{\partial \zeta^*} \right) \\
&= \sum_{i=1}^{H-1} \gamma_{u_i} \dot{\mathbf{v}}_i^* \\
&= \mathbf{0}.
\end{aligned}$$

For $\phi(\mathbf{z}; \mathbf{u})$ that is linear in \mathbf{u} , the partial derivative, $\frac{\partial \phi(\mathbf{z}; \mathbf{u})}{\partial u_n}$, is a function that does not involve \mathbf{u} . Thus, the gradient for \mathbf{u}_H is

$$\begin{aligned}
\dot{u}_{H,n} &= -\frac{1}{P} \sum_{\mu=1}^P \mathbf{v}_H^\top \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_H)^\top}{\partial u_{H,n}} \frac{\partial \ell_\mu}{\partial \zeta} \\
&= \sum_{i=1}^{H-1} \gamma_{v_i} \left(-\frac{1}{P} \sum_{\mu=1}^P \mathbf{v}_i^{*\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_H)^\top}{\partial u_{H,n}} \frac{\partial \ell_\mu}{\partial \zeta} \right) \\
&= \sum_{i=1}^{H-1} \gamma_{v_i} \dot{u}_{i,n}^* \\
&= 0, \quad n = 1, \dots, N_u.
\end{aligned}$$

Hence, in all the four cases, the weights of the width- H network given by Theorem 1 satisfy that the input-output map is the same as the width- $(H-1)$ network and the gradients are zero. \blacksquare

F Invariant Manifolds

F.1 Definition of invariant manifolds

We provide the definition of invariant manifolds used in this work.

Definition 2 (Invariant manifold). A set $\mathcal{M} \subset \mathbb{R}^n$ is an invariant set under the dynamical system $\dot{\theta} = h(\theta)$ if for any $\theta(0) = \theta_0 \in \mathcal{M}$ we have $\theta(t) \in \mathcal{M}$ for all $t \in \mathbb{R}$. An invariant set $\mathcal{M} \subset \mathbb{R}^n$ is an invariant manifold if \mathcal{M} has the structure of a differentiable manifold.

In this work, we focus on a particular class of invariant manifolds defined via the constraint $\Phi(\theta) = 0$; see Theorem 3. For defining invariant manifolds in more generality, we refer the readers to [Wiggins \(2003\)](#).

If a trajectory on an invariant manifold connects two distinct fixed points as $t \rightarrow \pm\infty$, it is called a heteroclinic orbit in the dynamical systems literature ([Bakhtin, 2011](#)). The saddle-to-saddle transitions in our setup can be viewed as heteroclinic orbits.

F.2 Proof of invariant manifolds

We here prove Theorem 3.

Proof. We prove the four statements one by one. Recall that θ_i is defined in Equation (1), which is the stacked second-layer and first-layer weights in the i -th unit

$$\theta_i = \begin{bmatrix} \mathbf{v}_i \\ \mathbf{u}_i \end{bmatrix}.$$

(i) For any ϕ , two units have equal weights: $\theta_i = \theta_j$.

We write down the dynamics of the difference $(\theta_i - \theta_j)$ and substitute in $\theta_i = \theta_j$. Using Equation (21), the dynamics of $(\mathbf{v}_i - \mathbf{v}_j)$ is given by

$$\frac{d}{dt}(\mathbf{v}_i - \mathbf{v}_j) = -\frac{1}{P} \sum_{\mu=1}^P (\phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i) - \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_j))^T \frac{\partial \ell_\mu}{\partial \zeta} = \mathbf{0}.$$

Using Equation (22), the dynamics of $(u_{i,n} - u_{j,n})$ for $n = 1, \dots, N_u$ is given by

$$\frac{d}{dt}(u_{i,n} - u_{j,n}) = -\frac{1}{P} \sum_{\mu=1}^P \left(\mathbf{v}_i^\top \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^\top}{\partial u_{i,n}} - \mathbf{v}_j^\top \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_j)^\top}{\partial u_{j,n}} \right) \frac{\partial \ell_\mu}{\partial \zeta} = 0.$$

(ii) If $\exists \mathbf{u}_{\text{zero}}$ such that $\forall z, \phi(z; \mathbf{u}_{\text{zero}}) = 0$, a unit has zero weights: $\mathbf{v}_i = \mathbf{0}, \mathbf{u}_i = \mathbf{u}_{\text{zero}}$.

Substituting $\mathbf{v}_i = \mathbf{0}, \mathbf{u}_i = \mathbf{u}_{\text{zero}}$ into Equations (21) and (22), we obtain

$$\begin{aligned} \dot{\mathbf{v}}_i &= -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_{\text{zero}})^\top \frac{\partial \ell_\mu}{\partial \zeta} = \mathbf{0}, \\ \dot{u}_{i,n} &= -\frac{1}{P} \sum_{\mu=1}^P \mathbf{0}^\top \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^\top}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \zeta} = 0, \quad n = 1, \dots, N_u. \end{aligned}$$

(iii) If $\phi(z; \mathbf{u})$ is homogeneous in \mathbf{u} , two units have proportional weights: $\theta_i = \gamma \theta_j, \gamma \in \mathbb{F}$.

Using Equation (21) and the degree-1 homogeneous property, $\forall \gamma \in \mathbb{F}, \phi(\mathbf{z}; \gamma \mathbf{u}) = \gamma \phi(\mathbf{z}; \mathbf{u})$, the dynamics of $(\mathbf{v}_i - \gamma \mathbf{v}_j)$ is given by

$$\begin{aligned} \frac{d}{dt}(\mathbf{v}_i - \gamma \mathbf{v}_j) &= -\frac{1}{P} \sum_{\mu=1}^P (\phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i) - \gamma \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_j))^{\top} \frac{\partial \ell_\mu}{\partial \zeta} \\ &= -\frac{1}{P} \sum_{\mu=1}^P (\phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i) - \phi(g_{\text{in}}(\mathbf{x}_\mu); \gamma \mathbf{u}_j))^{\top} \frac{\partial \ell_\mu}{\partial \zeta} \\ &= \mathbf{0}. \end{aligned}$$

Using Equations (22) and (25), the dynamics of $(u_{i,n} - \gamma u_{j,n})$ for $n = 1, \dots, N_u$ is given by

$$\begin{aligned} \frac{d}{dt}(u_{i,n} - \gamma u_{j,n}) &= -\frac{1}{P} \sum_{\mu=1}^P \left(\mathbf{v}_i^{\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^{\top}}{\partial u_{i,n}} - \gamma \mathbf{v}_j^{\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_j)^{\top}}{\partial u_{j,n}} \right) \frac{\partial \ell_\mu}{\partial \zeta} \\ &= -\frac{1}{P} \sum_{\mu=1}^P (\mathbf{v}_i - \gamma \mathbf{v}_j)^{\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^{\top}}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \zeta} \\ &= \mathbf{0}. \end{aligned}$$

(iv) If $\phi(\mathbf{z}; \mathbf{u})$ is linear in \mathbf{u} , any number of units have linear dependence: $\theta_i = \sum_{j \neq i} \gamma_j \theta_j$.

We leverage the degree-1 homogeneous and additive properties of linearity, which yields

$$\phi(\mathbf{z}; \mathbf{u}_i) - \sum_{j \neq i} \gamma_j \phi(\mathbf{z}; \mathbf{u}_j) = \phi \left(\mathbf{z}; \mathbf{u}_i - \sum_{j \neq i} \gamma_j \mathbf{u}_j \right) \quad (27)$$

Using Equations (21) and (27), we obtain the dynamics of $(\mathbf{v}_i - \sum_{j \neq i} \gamma_j \mathbf{v}_j)$

$$\begin{aligned} \frac{d}{dt} \left(\mathbf{v}_i - \sum_{j \neq i} \gamma_j \mathbf{v}_j \right) &= -\frac{1}{P} \sum_{\mu=1}^P \left(\phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i) - \sum_{j \neq i} \gamma_j \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_j) \right)^{\top} \frac{\partial \ell_\mu}{\partial \zeta} \\ &= -\frac{1}{P} \sum_{\mu=1}^P \phi \left(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i - \sum_{j \neq i} \gamma_j \mathbf{u}_j \right)^{\top} \frac{\partial \ell_\mu}{\partial \zeta} \\ &= -\frac{1}{P} \sum_{\mu=1}^P \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{0})^{\top} \frac{\partial \ell_\mu}{\partial \zeta} \\ &= \mathbf{0}. \end{aligned}$$

For $\phi(\mathbf{z}; \mathbf{u})$ that is linear in \mathbf{u} , the partial derivative $\frac{\partial \phi(\mathbf{z}; \mathbf{u})}{\partial u_n}$ is a function that does not involve \mathbf{u} . Thus, using Equation (22), we obtain the dynamics of $(u_{i,n} - \sum_{j \neq i} \gamma_j u_{j,n})$ for $n = 1, \dots, N_u$

$$\frac{d}{dt} \left(u_{i,n} - \sum_{j \neq i} \gamma_j u_{j,n} \right) = -\frac{1}{P} \sum_{\mu=1}^P \left(\mathbf{v}_i - \sum_{j \neq i} \gamma_j \mathbf{v}_j \right)^{\top} \frac{\partial \phi(g_{\text{in}}(\mathbf{x}_\mu); \mathbf{u}_i)^{\top}}{\partial u_{i,n}} \frac{\partial \ell_\mu}{\partial \zeta} = 0.$$

■

E.3 Expressivity on invariant manifolds

When the weights of a network lie on an invariant manifolds, its input-output map is expressible by the architecture with fewer units than its actual width.

- (i) For any ϕ , when two units have equal weights, $\theta_i = \theta_j$, we can remove the i -th unit and multiply v_j by 2, obtaining a network with one less unit that expresses the same input-output map.
- (ii) For ϕ such that $\forall z, \phi(z; u_{\text{zero}}) = 0$, when a unit has zero weights, $v_i = \mathbf{0}, u_i = u_{\text{zero}}$, we can just remove this unit, obtaining a network with one less unit that expresses the same input-output map.
- (iii) For $\phi(z; u)$ that is degree-1 homogeneous in u , when two units have proportional weights, $\theta_i = \gamma \theta_j$, we can remove the i -th unit and multiply v_j by $(1 + \gamma^2)$, obtaining a network with one less unit that expresses the same input-output map.
- (iv) For $\phi(z; u)$ that is linear in u , when there is a linear dependence $\theta_i = \sum_{j \neq i} \gamma_j \theta_j$, we can remove the i -th unit and modify the second-layer weights in all remaining units as follows

$$v_j^{\text{new}} = v_j + \gamma_j \sum_{j' \neq i} \gamma_{j'} v_{j'}. \quad (28)$$

This new width- $(H - 1)$ network expresses the same input-output map as the original width- H network

$$\begin{aligned} \sum_{j=1, j \neq i}^H \phi(g_{\text{in}}(\mathbf{x}); u_j) v_j^{\text{new}} &= \sum_{j \neq i} \phi(g_{\text{in}}(\mathbf{x}); u_j) \left(v_j + \gamma_j \sum_{j' \neq i} \gamma_{j'} v_{j'} \right) \\ &= \sum_{j \neq i} \phi(g_{\text{in}}(\mathbf{x}); u_j) v_j + \phi \left(g_{\text{in}}(\mathbf{x}); \sum_{j \neq i} \gamma_j u_j \right) \sum_{j' \neq i} \gamma_{j'} v_{j'} \\ &= \sum_{j \neq i} \phi(g_{\text{in}}(\mathbf{x}); u_j) v_j + \phi(g_{\text{in}}(\mathbf{x}); u_i) v_i \\ &= \sum_{j=1}^H \phi(g_{\text{in}}(\mathbf{x}); u_j) v_j. \end{aligned}$$

F.4 The embedded fixed points on invariant manifolds

The set of embedded fixed points lying on the invariant manifolds given in Theorem 3 is a subset of the embedded fixed points given in Theorem 1. Here we specify the subset of embedded fixed points that lie on invariant manifolds.

- (i) When $\gamma_v = 1/2$ in Equation (4), the embedded fixed point has $u_H = u_i = u_i^*, v_H = v_i = v_i^*/2$, which is on the invariant manifold of $\theta_H = \theta_i$ in Theorem 3(i).
- (ii) For ϕ such that $\forall z, \phi(z; u_{\text{zero}}) = 0$, it is clear that $u_H = u_{\text{zero}}, v_H = \mathbf{0}$ is on the invariant manifold in Theorem 3(ii).
- (iii) For $\phi(z; u)$ that is degree-1 homogeneous in u , when $\gamma_v = \gamma_u/(1 + \gamma_u^2)$ in Equation (6), the embedded fixed point has

$$u_H = \gamma_u u_i, v_H = \frac{\gamma_v}{1 - \gamma_u \gamma_v} v_i = \gamma_u v_i, \quad (29)$$

which is on the invariant manifold of $\theta_H = \gamma_u \theta_i$ in Theorem 3(iii).

- (iv) For $\phi(z; u)$ that is linear in u , let us first rearrange Equation (7) by substituting $v_i = v_i^* - \gamma_{u_i} v_H$ into $v_H = \sum_{i=1}^{H-1} \gamma_{v_i} v_i^*$ and obtaining an expression of v_H in terms of $\{v_i\}_{i=1}^{H-1}$

$$v_H = \sum_{i=1}^{H-1} \gamma_{v_i} (v_i + \gamma_{u_i} v_H) \Rightarrow v_H = \frac{1}{1 - \sum_{j=1}^{H-1} \gamma_{v_j} \gamma_{u_j}} \sum_{i=1}^{H-1} \gamma_{v_i} v_i. \quad (30)$$

When $\gamma_{v_i} = \gamma_{u_i}/(1 + \sum_{j=1}^{H-1} \gamma_{u_j}^2)$ in Equation (7), the embedded fixed point has

$$\mathbf{u}_H = \sum_{i=1}^{H-1} \gamma_{u_i} \mathbf{u}_i, \quad (31a)$$

$$\begin{aligned} \mathbf{v}_H &= \frac{1}{1 - \sum_{j=1}^{H-1} \gamma_{v_j} \gamma_{u_j}} \sum_{i=1}^{H-1} \gamma_{v_i} \mathbf{v}_i \\ &= \frac{1}{1 - \sum_{j=1}^{H-1} \frac{\gamma_{u_j}^2}{1 + \sum_{k=1}^{H-1} \gamma_{u_k}^2}} \sum_{i=1}^{H-1} \frac{\gamma_{u_i}}{1 + \sum_{j=1}^{H-1} \gamma_{u_j}^2} \mathbf{v}_i \\ &= \sum_{i=1}^{H-1} \gamma_{u_i} \mathbf{v}_i, \end{aligned} \quad (31b)$$

which is on the invariant manifold of $\boldsymbol{\theta}_H = \sum_{i=1}^{H-1} \gamma_{u_i} \boldsymbol{\theta}_i$ in Theorem 3(iv).

G Dynamics of Linear Networks

G.1 Gradient flow equations

We derive the gradient flow equations given in Equation (9).

For $i = 1, \dots, H$, the gradient flow dynamics on squared loss, $\ell(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{2} \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$, is given by

$$\begin{aligned}\dot{\mathbf{v}}_i &= -\frac{1}{P} \sum_{\mu=1}^P \frac{\partial \mathcal{L}}{\partial f(\mathbf{x}_\mu)} \frac{\partial f(\mathbf{x}_\mu)}{\partial \mathbf{v}_i} \\ &= \frac{1}{P} \sum_{\mu=1}^P (\mathbf{y}_\mu - \mathbf{W} \mathbf{z}_\mu) \mathbf{z}_\mu^\top \mathbf{u}_i \\ &= \left(\frac{1}{P} \sum_{\mu=1}^P \mathbf{y}_\mu \mathbf{z}_\mu^\top - \mathbf{W} \frac{1}{P} \sum_{\mu=1}^P \mathbf{z}_\mu \mathbf{z}_\mu^\top \right) \mathbf{u}_i, \\ \dot{\mathbf{u}}_i &= -\frac{1}{P} \sum_{\mu=1}^P \frac{\partial \mathcal{L}}{\partial f(\mathbf{x}_\mu)} \frac{\partial f(\mathbf{x}_\mu)}{\partial \mathbf{u}_i} \\ &= \frac{1}{P} \sum_{\mu=1}^P \mathbf{z}_\mu (\mathbf{y}_\mu - \mathbf{W} \mathbf{z}_\mu)^\top \mathbf{v}_i \\ &= \left(\frac{1}{P} \sum_{\mu=1}^P \mathbf{z}_\mu \mathbf{y}_\mu^\top - \frac{1}{P} \sum_{\mu=1}^P \mathbf{z}_\mu \mathbf{z}_\mu^\top \mathbf{W}^\top \right) \mathbf{v}_i.\end{aligned}$$

Recall that the data statistics are defined as

$$\Sigma_{yz} = \frac{1}{P} \sum_{\mu=1}^P \mathbf{y}_\mu \mathbf{z}_\mu^\top, \quad \Sigma_{zz} = \frac{1}{P} \sum_{\mu=1}^P \mathbf{z}_\mu \mathbf{z}_\mu^\top.$$

Substituting in the data statistics, we obtain the gradient flow equations in Equation (9), which are

$$\dot{\mathbf{v}}_i = (\Sigma_{yz} - \mathbf{W} \Sigma_{zz}) \mathbf{u}_i, \quad \dot{\mathbf{u}}_i = (\Sigma_{yz} - \mathbf{W} \Sigma_{zz})^\top \mathbf{v}_i, \quad i = 1, \dots, H.$$

G.2 Proof of timescale separation

We here prove Theorem 4, that is the timescale separation between directions in a linear dynamical system.

Proof. The linear dynamical system in Equation (10) can be written as

$$\dot{\boldsymbol{\theta}}_i = \mathbf{M} \boldsymbol{\theta}_i, \quad \text{where } \mathbf{M} = \begin{bmatrix} \mathbf{0} & \Sigma_{yx} \\ \Sigma_{yx}^\top & \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\theta}_i = \begin{bmatrix} \mathbf{v}_i \\ \mathbf{u}_i \end{bmatrix}. \quad (32)$$

The symmetric matrix \mathbf{M} has D positive eigenvalues, D negative eigenvalues, and $(N_v + N_u - 2D)$ zero eigenvalues. The nonzero eigenvalues are the singular values of Σ_{yx} and their negative

$$\mathbf{M} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{r}_k \end{bmatrix} = s_k \begin{bmatrix} \mathbf{q}_k \\ \mathbf{r}_k \end{bmatrix}, \quad \mathbf{M} \begin{bmatrix} \mathbf{q}_k \\ -\mathbf{r}_k \end{bmatrix} = -s_k \begin{bmatrix} \mathbf{q}_k \\ -\mathbf{r}_k \end{bmatrix}, \quad k = 1, \dots, D. \quad (33)$$

The exact time-course solution to Equation (10) is

$$\boldsymbol{\theta}_i(t) = \begin{bmatrix} \mathbf{v}_i \\ \mathbf{u}_i \end{bmatrix}(t) = \sum_{k=1}^D \left(c_{ki} e^{s_k t} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{r}_k \end{bmatrix} + b_{ki} e^{-s_k t} \begin{bmatrix} \mathbf{q}_k \\ -\mathbf{r}_k \end{bmatrix} \right) + \boldsymbol{\xi}_i, \quad i = 1, \dots, H, \quad (34)$$

where the constants c_{ki}, b_{ki} are projections of the initial weights onto the eigenvectors with nonzero eigenvalues, and ξ_i is the initial weights projected onto the eigenspace with zero eigenvalue

$$c_{ki} = \frac{1}{2} (\mathbf{q}_k^\top \mathbf{v}_i(0) + \mathbf{r}_k^\top \mathbf{u}_i(0)), \quad (35a)$$

$$b_{ki} = \frac{1}{2} (\mathbf{q}_k^\top \mathbf{v}_i(0) - \mathbf{r}_k^\top \mathbf{u}_i(0)), \quad (35b)$$

$$\xi_i = \boldsymbol{\theta}_i(0) - \sum_{k=1}^D \left(c_{ki} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{r}_k \end{bmatrix} + b_{ki} \begin{bmatrix} \mathbf{q}_k \\ -\mathbf{r}_k \end{bmatrix} \right). \quad (35c)$$

Recall that the projection matrix \mathbf{P} , defined in Equation (11), corresponds to the rank- r subspace spanned by the top r singular vectors. The projection of the weights on this subspace has ℓ_2 norm

$$\|\mathbf{P}\boldsymbol{\theta}_i\| = \left\| \sum_{k=1}^r c_{ki} e^{s_k t} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{r}_k \end{bmatrix} \right\| = e^{s_1 t} \sqrt{2 \sum_{k=1}^r c_{ki}^2} = e^{s_1 t} \|\mathbf{P}\boldsymbol{\theta}_i(0)\| \quad (36)$$

The time it takes for $\|\mathbf{P}\boldsymbol{\theta}_i\|$ to reach $O(1)$ is

$$T = \frac{1}{s_1} \ln \frac{1}{\|\mathbf{P}\boldsymbol{\theta}_i(0)\|}. \quad (37)$$

At time T , the projection of the weights on the null-space of \mathbf{P} has ℓ_2 norm

$$\begin{aligned} \|(\mathbf{I} - \mathbf{P})\boldsymbol{\theta}_i(T)\| &= \left\| \sum_{k=r+1}^D c_{ki} e^{s_k T} \begin{bmatrix} \mathbf{q}_k \\ \mathbf{r}_k \end{bmatrix} + \sum_{k=1}^D b_{ki} e^{-s_k T} \begin{bmatrix} \mathbf{q}_k \\ -\mathbf{r}_k \end{bmatrix} + \xi_i \right\| \\ &= \sqrt{2 \sum_{k=r+1}^D c_{ki}^2 e^{2s_k T} + 2 \sum_{k=1}^D b_{ki}^2 e^{-2s_k T} + \|\xi_i\|^2} \\ &\leq e^{s_{r+1} T} \sqrt{2 \sum_{k=r+1}^D c_{ki}^2 + 2 \sum_{k=1}^D b_{ki}^2 + \|\xi_i\|^2} \\ &= \left(\frac{1}{\|\mathbf{P}\boldsymbol{\theta}_i(0)\|} \right)^{\frac{s_{r+1}}{s_1}} \|(\mathbf{I} - \mathbf{P})\boldsymbol{\theta}_i(0)\| \\ &= \left(\frac{\|(\mathbf{I} - \mathbf{P})\boldsymbol{\theta}_i(0)\|}{\|\mathbf{P}\boldsymbol{\theta}_i(0)\|} \right)^{\frac{s_{r+1}}{s_1}} \|(\mathbf{I} - \mathbf{P})\boldsymbol{\theta}_i(0)\|^{1 - \frac{s_{r+1}}{s_1}} \end{aligned} \quad (38)$$

Because each entry of the initial weight $\boldsymbol{\theta}_i(0)$ is independently sampled from $\mathcal{N}(0, \epsilon^2)$, the norm is $\|\boldsymbol{\theta}_i(0)\| = O(\epsilon)$ and the ratio $\frac{\|(\mathbf{I} - \mathbf{P})\boldsymbol{\theta}_i(0)\|}{\|\mathbf{P}\boldsymbol{\theta}_i(0)\|} = O(1)$. Thus, we have

$$\|(\mathbf{I} - \mathbf{P})\boldsymbol{\theta}_i(T)\| = O(1) \|(\mathbf{I} - \mathbf{P})\boldsymbol{\theta}_i(0)\|^{1 - \frac{s_{r+1}}{s_1}} = O\left(\epsilon^{1 - \frac{s_{r+1}}{s_1}}\right). \quad (39)$$

Hence, at time T , $\|\mathbf{P}\boldsymbol{\theta}_i(T)\| = O(1)$, while $\|(\mathbf{I} - \mathbf{P})\boldsymbol{\theta}_i(T)\| = O(\epsilon^{1 - s_{r+1}/s_1})$ is still small. ■

G.3 Fixed points of linear networks

In Theorem 6, we specify all the fixed points in the linear network learning dynamics in Equation (9).

Lemma 6. Denote the eigenvectors of the symmetric matrix $\Sigma_{yz}\Sigma_{zz}^{-1}\Sigma_{yz}^\top$ as $\mathbf{e}_k \in \mathbb{R}^{N_v}$, $k = 1, \dots, N_v$, arranged in descending order of their associated eigenvalues. There are at most $D = \min(N_v, N_u)$ nonzero eigenvalues. The sufficient and necessary condition for a set of weights to be a fixed point of the dynamics in Equation (9) is

$$\sum_{i=1}^H \mathbf{v}_i \mathbf{u}_i^\top = \sum_{k \in \mathcal{A}_r} \mathbf{e}_k \mathbf{e}_k^\top \Sigma_{yz} \Sigma_{zz}^{-1}, \quad \mathbf{v}_i \in \text{span}\{\mathbf{e}_k\}_{k \in \mathcal{A}_r}, \quad \mathbf{u}_i \in \text{span}\{\Sigma_{zz}^{-1} \Sigma_{yz}^\top \mathbf{e}_k\}_{k \in \mathcal{A}_r}, \quad (40)$$

where \mathcal{A}_r is a set of indices $\mathcal{A}_r \subseteq \{1, 2, \dots, D\}$, $|\mathcal{A}_r| = r$.

Proof. The proof can be found in the seminal work by [Baldi & Hornik \(1989\)](#) or follow-up work ([Kawaguchi, 2016](#); [Lu & Kawaguchi, 2017](#); [Yun et al., 2018](#); [Laurent & von Brecht, 2018](#); [Achour et al., 2024](#)). ■

The dynamics near a fixed points defined in Equation (40) is approximately a linear dynamical system, for $i = 1, \dots, H$,

$$\dot{\mathbf{u}}_i = (\Sigma_{yz} - \mathbf{W}\Sigma_{zz})\mathbf{u}_i \approx \left(\Sigma_{yz} - \sum_{k \in \mathcal{A}_r} \mathbf{e}_k \mathbf{e}_k^\top \Sigma_{yz} \Sigma_{zz}^{-1} \Sigma_{zz} \right) \mathbf{u}_i = \tilde{\Sigma}_{yz} \mathbf{u}_i, \quad (41a)$$

$$\dot{\mathbf{v}}_i = (\Sigma_{yz} - \mathbf{W}\Sigma_{zz})^\top \mathbf{v}_i \approx \left(\Sigma_{yz} - \sum_{k \in \mathcal{A}_r} \mathbf{e}_k \mathbf{e}_k^\top \Sigma_{yz} \Sigma_{zz}^{-1} \Sigma_{zz} \right)^\top \mathbf{v}_i = \tilde{\Sigma}_{yz}^\top \mathbf{v}_i. \quad (41b)$$

where $\tilde{\Sigma}_{yz}$ is Σ_{yz} projected onto a rank- $(D - r)$ subspace

$$\tilde{\Sigma}_{yz} = \sum_{k \notin \mathcal{A}_r} \mathbf{e}_k \mathbf{e}_k^\top \Sigma_{yz}. \quad (42)$$

Remark 2. When defining \mathcal{A}_r in Theorem 6, there are $\binom{D}{r}$ possible choices of r indices out of D , assuming the eigenvalues are distinct. Each choice produces a different input-output linear map. Thus, there are $\binom{D}{r}$ embedded fixed points of effective width r in linear networks, if we count fixed points by the distinct input-output maps they implement. When a linear network undergoes saddle-to-saddle dynamics and approaches fixed points of effective width $r = 0, 1, \dots, D$ sequentially, determining which of the $\binom{D}{r}$ fixed points it approaches is a non-trivial open problem. Even in diagonal linear networks, specifying the sequence of visited saddles requires non-trivial work ([Berthier, 2023](#); [Pesme & Flammarion, 2023](#)).

H Dynamics of Quadratic Networks

H.1 Gradient flow equations

The gradient flow dynamics of Equation (13) trained on squared loss, $\ell(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$, is given by

$$\begin{aligned}\dot{v}_i &= -\frac{1}{P} \sum_{\mu=1}^P \frac{\partial \mathcal{L}}{\partial f(\mathbf{x}_\mu)} \frac{\partial f(\mathbf{x}_\mu)}{\partial v_i} \\ &= \frac{1}{P} \sum_{\mu=1}^P \left(y_\mu - \sum_{j=1}^H v_j \mathbf{u}_j^\top \mathbf{Z}_\mu \mathbf{u}_j \right) \mathbf{u}_i^\top \mathbf{Z}_\mu \mathbf{u}_i \\ &= \mathbf{u}_i^\top \left(\frac{1}{P} \sum_{\mu=1}^P y_\mu \mathbf{Z}_\mu \right) \mathbf{u}_i - \sum_{j=1}^H v_j \mathbf{u}_j^\top \left(\frac{1}{P} \sum_{\mu=1}^P \mathbf{Z}_\mu \mathbf{u}_j \mathbf{u}_i^\top \mathbf{Z}_\mu \right) \mathbf{u}_i, \\ \dot{\mathbf{u}}_i &= -\frac{1}{P} \sum_{\mu=1}^P \frac{\partial \mathcal{L}}{\partial f(\mathbf{x}_\mu)} \frac{\partial f(\mathbf{x}_\mu)}{\partial \mathbf{u}_i} \\ &= \frac{1}{P} \sum_{\mu=1}^P \left(y_\mu - \sum_{j=1}^H v_j \mathbf{u}_j^\top \mathbf{Z}_\mu \mathbf{u}_j \right) 2\mathbf{Z}_\mu \mathbf{u}_i \\ &= 2\mathbf{u}_i^\top \left(\frac{1}{P} \sum_{\mu=1}^P y_\mu \mathbf{Z}_\mu \right) \mathbf{u}_i - 2 \sum_{j=1}^H v_j \left(\frac{1}{P} \sum_{\mu=1}^P \mathbf{Z}_\mu \mathbf{u}_i \mathbf{u}_j^\top \mathbf{Z}_\mu \right) \mathbf{u}_j.\end{aligned}$$

Denote the data statistics as

$$\Sigma_{yZ} = \frac{1}{P} \sum_{\mu=1}^P y_\mu \mathbf{Z}_\mu, \quad \Sigma_{ZZ} = \frac{1}{P} \sum_{\mu=1}^P \text{vec}(\mathbf{Z}_\mu) \text{vec}(\mathbf{Z}_\mu)^\top. \quad (43)$$

The dynamics can be written as

$$\dot{v}_i = \mathbf{u}_i^\top \Sigma_{yZ} \mathbf{u}_i - \sum_{j=1}^H v_j (\mathbf{u}_j \otimes \mathbf{u}_j)^\top \Sigma_{ZZ} (\mathbf{u}_i \otimes \mathbf{u}_i), \quad (44a)$$

$$\dot{\mathbf{u}}_i = 2v_i \Sigma_{yZ} \mathbf{u}_i - 2v_i \sum_{j=1}^H v_j (\mathbf{u}_j \otimes \mathbf{u}_j)^\top \Sigma_{ZZ} (\mathbf{u}_i \otimes \mathbf{I}_D), \quad (44b)$$

where \otimes denotes the Kronecker product.

H.2 Derivations for timescale separation

We here provide the derivations for Theorem 5, that is the timescale separation between units in quadratic dynamics.

To reduce clutter, we omit the index i in Equation (14) for now; we will put it back when we need it. We study the dynamics

$$\dot{v} = \mathbf{u}^\top \Sigma_{yZ} \mathbf{u}, \quad \dot{\mathbf{u}} = 2v \Sigma_{yZ} \mathbf{u}. \quad (45)$$

Step 1: reduction to one-dimensional dynamics.

Denote the eigenvalues and eigenvectors of the symmetric matrix Σ_{yZ} as

$$\Sigma_{yZ} \mathbf{r}_k = s_k \mathbf{r}_k, \quad k = 1, \dots, D. \quad (46)$$

We change variables by projecting \mathbf{u} onto the (orthonormal) eigenvectors of Σ_{yZ} ,

$$a_k \equiv \frac{1}{\sqrt{2}} \mathbf{r}_k^\top \mathbf{u}, \quad k = 1, \dots, D \quad (47)$$

where the factor of $\sqrt{2}$ is for convenience only. Since time has arbitrary unit, we can let $t \rightarrow t/2$. Then, the dynamics of \mathbf{u} and v can be expressed in terms of the new coordinates a_1, \dots, a_D and v ,

$$\dot{v} = \sum_{k=1}^D s_k a_k^2, \quad (48a)$$

$$\dot{a}_k = v s_k a_k, \quad k = 1, \dots, D. \quad (48b)$$

This set of equations admits a conservation law,

$$\frac{d}{dt} \left(v^2 - \sum_{k=1}^D a_k^2 \right) = 2v \sum_{k=1}^D s_k a_k^2 - 2v \sum_{k=1}^D s_k a_k^2 = 0. \quad (49)$$

Thus, their difference at initialization is conserved throughout training

$$v(t)^2 - \sum_{k=1}^D a_k(t)^2 = v(0)^2 - \sum_{k=1}^D a_k(0)^2. \quad (50)$$

We also notice a relationship between the a_k ,

$$\frac{da_m}{da_k} = \frac{s_m a_m}{s_k a_k} \Rightarrow \frac{1}{s_m} \ln \frac{a_m(t)}{a_m(0)} = \frac{1}{s_k} \ln \frac{a_k(t)}{a_k(0)}. \quad (51)$$

Thus, different $a_k(t)$ can be expressed in terms of each other,

$$a_k(t) = a_k(0) \left(\frac{a_m(t)}{a_m(0)} \right)^{s_k/s_m}. \quad (52)$$

Using Equations (50) and (52), and defining

$$\pi_k(t) \equiv \frac{a_k(t)}{a_k(0)}, \quad (53)$$

we can express $v(t)$ in terms of $\pi_m(t)$,

$$v(t)^2 = v(0)^2 + \sum_{k=1}^D a_k(0)^2 (\pi_m(t)^{2s_k/s_m} - 1). \quad (54)$$

At this point m is arbitrary; we will set it to a particular value shortly. Substituting Equation (54) into Equation (48b) and using Equation (53), we obtain a one-dimensional and separable differential equation for π_m

$$\dot{\pi}_m = v s_m \pi_m = \text{sign}(v(0)) s_m \pi_m \sqrt{v(0)^2 + \sum_{k=1}^D a_k(0)^2 (\pi_m^{2s_k/s_m} - 1)} \quad (55)$$

with initial condition $\pi_m(0) = 1$. Reducing the $(D+1)$ -dimensional dynamics in Equation (45) to the one-dimensional separable differential equation in Equation (55) may be of independent interest.

Step 2: bounding the growth time.

Let us choose m to maximize $\text{sign}(v(0)) s_m$. If $v(0)$ is positive, the chosen m maximizes s_m ; if it is negative the chosen m minimizes s_m (and thus maximizes $|s_m|$, assuming that there are both negative and positive eigenvalues). In either case, $\max_k(s_k/s_m) = 1$. And with this maximization, the prefactor becomes $|s_m|$.

We are interested in the time it takes for $a_m(t)$ to become large; that happens when $t \sim 1/a_m(0)$, which is large for small initial conditions. The time it takes for that to happen, denoted t_{final} , is bounded by the time it takes for $\pi_m(t)$ to go to infinity, giving us

$$t_{\text{final}} < t_\infty = \frac{1}{|s_m|} \int_1^\infty \frac{d\pi}{\pi \sqrt{v(0)^2 + \sum_{k=1}^D a_k(0)^2 (\pi^{2s_k/s_m} - 1)}}. \quad (56)$$

So far we have ignored the dependence on unit, i . However, each i has associated with it a different t_∞ . Let us use $t_{\infty,i}$ to denote the different times, arranged in increasing order,

$$t_{\infty,1} < t_{\infty,2} < \dots < t_{\infty,H}. \quad (57)$$

We will also add a subscript i to all the other variables as well. At time $t_{\infty,1}$, we know that $\pi_{m,1}(t_{\infty,1})$ is large, but what about $\pi_{m,i}(t_{\infty,1})$ for $i > 1$? That is given implicitly by

$$t_{\text{final}} = \frac{1}{|s_{m,i}|} \int_1^{\pi_{m,i}(t_{\text{final}})} \frac{d\pi}{\pi \sqrt{v_i(0)^2 + \sum_{k=1}^D a_{k,i}(0)^2 (\pi^{2s_k/s_{m,i}} - 1)}}. \quad (58)$$

Note that s_m acquired a subscript i , because it is either the maximum or minimum eigenvalue, depending on the sign of $v_i(0)$. Rearranging terms slightly and performing a small amount of algebra, this can be written as

$$\int_{\pi_{m,i}(t_{\text{final}})}^{\infty} \frac{d\pi}{\pi^2 \sqrt{1 + \Psi_i(\pi)}} = |s_{m,i} a_{m,i}(0)| (t_{\infty,i} - t_{\text{final}}) \quad (59)$$

where

$$\Psi_i(\pi) \equiv \frac{1}{\pi^2} \left(\frac{v_i(0)^2}{a_{m,i}(0)^2} + \sum_{k \neq m} \frac{a_{k,i}(0)^2}{a_{m,i}(0)^2} (\pi^{2s_k/s_{m,i}} - 1) - 1 \right). \quad (60)$$

We can bound the left hand side,

$$\int_{\pi_{m,i}(t_{\text{final}})}^{\infty} \frac{d\pi}{\pi^2 \sqrt{1 + \Psi_i(\pi)}} < \frac{1}{\sqrt{1 + \Psi_{\min,i}}} \int_{\pi_{m,i}(t_{\text{final}})}^{\infty} \frac{d\pi}{\pi^2} = \frac{1}{\sqrt{1 + \Psi_{\min,i}}} \frac{1}{\pi_{m,i}(t_{\text{final}})} \quad (61)$$

where $\Psi_{\min,i}$ is the minimum value of $\Psi_i(\pi)$. Inserting this into Equation (59) then gives us

$$\pi_{m,i}(t_{\text{final}}) < \frac{1}{\sqrt{1 + \Psi_{\min,i}}} \frac{1}{|s_{m,i} a_{m,i}(0)| (t_{\infty,i} - t_{\text{final}})}. \quad (62)$$

To determine the size of the right hand side, we first note that because the spread in initial conditions is $O(1)$, the relative spread in the $t_{\infty,i}$ is $O(1)$. Second, the $t_{\infty,i}$ scale inversely with initial conditions (see Equation (56)), whose typical size we denote ϵ . Consequently,

$$|s_{m,i} a_{m,i}(0)| (t_{\infty,i} - t_{\text{final}}) \sim \epsilon \cdot \frac{1}{\epsilon} \sim O(1), \quad (63)$$

from which it follows that $\pi_{m,i}(t_{\text{final}}) \sim O(1)$. And so, given the definition of $\pi_m(t)$ in Equation (53), we have $a_{m,i}(t_{\text{final}}) \sim \epsilon$. Thus, when the variables associated with one of the units become $O(1)$, the variables associated with all the other units are $O(\epsilon)$.

I Implementation Details

Videos of the learning dynamics in Figure 1 are provided in the supplementary material.

For all models, we sample the initial weights from $\mathcal{N}(0, \epsilon^2)$ and train them with squared loss

$$\mathcal{L} = \frac{1}{P} \sum_{\mu=1}^P \ell(\mathbf{y}_\mu, f(\mathbf{x}_\mu)) = \frac{1}{P} \sum_{\mu=1}^P \|\mathbf{y}_\mu - f(\mathbf{x}_\mu)\|_2^2. \quad (64)$$

Linear fully-connected network (Figure 1B).

The network is defined as

$$f(\mathbf{x}) = \sum_{i=1}^H \mathbf{v}_i \mathbf{u}_i^\top \mathbf{x}, \quad \text{where } \mathbf{v}_i, \mathbf{u}_i, \mathbf{x} \in \mathbb{R}^2, H = 50. \quad (65)$$

The training set $\{\mathbf{x}_\mu, \mathbf{y}_\mu\}_{\mu=1}^P$ is generated as

$$\mathbf{y}_\mu = \mathbf{W}^* \mathbf{x}_\mu, \quad \mathbf{x}_\mu \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix}\right).$$

Here $P = 8192$, $\epsilon = 10^{-6}$, and the learning rate is 0.01.

Linear convolutional network (Figure 1C).

The network is defined as

$$f(\mathbf{x}) = \sum_{i=1}^H [v_{i1} \ v_{i2}] \begin{bmatrix} u_{i1} & u_{i2} & 0 & 0 \\ 0 & 0 & u_{i1} & u_{i2} \end{bmatrix} \mathbf{x}, \quad \text{where } \mathbf{x} \in \mathbb{R}^4, H = 50. \quad (66)$$

Here the first layer is a one-dimensional convolutional layer with kernel size 2, stride 2, and padding 0. We set $H = 50$. The pytorch code for setting up this layer is

```
torch.nn.Conv1d(in_channels=1,
                out_channels=50,
                kernel_size=2,
                stride=2,
                padding=0,
                dilation=1,
                groups=1,
                bias=False)
```

The training set $\{\mathbf{x}_\mu, \mathbf{y}_\mu\}_{\mu=1}^P$ is generated as

$$\mathbf{y}_\mu = \mathbf{w}^{*\top} \mathbf{x}_\mu, \quad \mathbf{x}_\mu \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right), \quad \mathbf{w}^* = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}.$$

Here $P = 8192$, $\epsilon = 10^{-6}$, and the learning rate is 0.01.

ReLU fully-connected network (Figure 1D).

The network is defined as

$$f(\mathbf{x}) = \sum_{i=1}^H v_i \text{ReLU}(\mathbf{u}_i^\top \mathbf{x}), \quad \text{where } v_i \in \mathbb{R}, \mathbf{u}_i, \mathbf{x} \in \mathbb{R}^2, H = 50. \quad (67)$$

The training set is an orthogonal input dataset used in [Boursier et al. \(2022, Figure 3\)](#) and [Zhang et al. \(2025a, Figure 4\)](#). It contains two data points

$$\begin{aligned} \mathbf{x}_1 &= \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}, & y_1 &= 1, \\ \mathbf{x}_2 &= \begin{bmatrix} -1 \\ 2 \end{bmatrix}, & y_2 &= -1. \end{aligned}$$

Here $P = 2$, $\epsilon = 10^{-6}$, and the learning rate is 0.01.

ReLU convolutional network (Figure 1E).

The network is defined as

$$f(\mathbf{x}) = \sum_{i=1}^H [v_{i1} \quad v_{i2}] \text{ReLU} \left(\begin{bmatrix} u_{i1} & u_{i2} & 0 & 0 \\ 0 & 0 & u_{i1} & u_{i2} \end{bmatrix} \mathbf{x} \right), \quad \text{where } \mathbf{x} \in \mathbb{R}^4, H = 50, \quad (68)$$

which is the same as Equation (66) except for the ReLU activation function. The training set $\{\mathbf{x}_\mu, y_\mu\}_{\mu=1}^P$ is generated as

$$y_\mu = \mathbf{w}^{*\top} \mathbf{x}_\mu, \quad \mathbf{w}^* = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}.$$

It contains four data points

$$\mathbf{x}_1 = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}_2 = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}_3 = \begin{bmatrix} 0 \\ 0 \\ 2\sqrt{2} \\ 0 \end{bmatrix}, \quad \mathbf{x}_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix}.$$

Here $P = 4$, $\epsilon = 10^{-6}$, and the learning rate is 0.01.

Linear self-attention (Figure 1F).

The model is defined as

$$f(\mathbf{X}) = \mathbf{X} + \sum_{i=1}^H \mathbf{V}_i \mathbf{X} \mathbf{X}^\top \mathbf{K}_i^\top \mathbf{Q}_i \mathbf{X}, \quad (69)$$

where

$$\mathbf{V}_i \in \mathbb{R}^{(D+1) \times (D+1)}, \quad \mathbf{K}_i, \mathbf{Q}_i \in \mathbb{R}^{R \times (D+1)}, \quad \mathbf{X} \in \mathbb{R}^{(D+1) \times (N+1)}. \quad (70)$$

Here D is the embedding dimension, N is the context length, and R is the rank of each attention head. We train the linear self-attention model on an in-context linear regression task (Garg et al., 2022; Zhang et al., 2025b). The training set $\{\mathbf{X}_\mu, y_{\mu,q}\}_{\mu=1}^P$ is generated as

$$\mathbf{X}_\mu = \begin{bmatrix} \mathbf{x}_{\mu,1} & \mathbf{x}_{\mu,2} & \cdots & \mathbf{x}_{\mu,N} & \mathbf{x}_{\mu,q} \\ \mathbf{w}_\mu^\top \mathbf{x}_{\mu,1} & \mathbf{w}_\mu^\top \mathbf{x}_{\mu,2} & \cdots & \mathbf{w}_\mu^\top \mathbf{x}_{\mu,N} & 0 \end{bmatrix}, \quad y_{\mu,q} = \mathbf{w}_\mu^\top \mathbf{x}_{\mu,q},$$

and

$$\mathbf{x}_{\mu,n}, \mathbf{x}_{\mu,q} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{w}_\mu \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad n = 1, \dots, N, \quad \mu = 1, \dots, P.$$

Here $D = 2$, $N = 32$, $R = 1$, $H = 10$, $P = 8192$, $\epsilon = 0.005$, and the learning rate is 0.02.

Quadratic network (Figure 1G).

The network is defined as

$$f(\mathbf{x}) = \sum_{i=1}^H v_i (\mathbf{u}_i^\top \mathbf{x})^2, \quad \text{where } v_i \in \mathbb{R}, \mathbf{u}_i, \mathbf{x} \in \mathbb{R}^2. \quad (71)$$

The training set $\{\mathbf{x}_\mu, y_\mu\}_{\mu=1}^P$ is generated as

$$y_\mu = (\mathbf{w}_1^{*\top} \mathbf{x}_\mu)^2 + (\mathbf{w}_2^{*\top} \mathbf{x}_\mu)^2, \quad \mathbf{x}_\mu \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad \mathbf{w}_1^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{w}_2^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Here $H = 10$, $P = 8192$, $\epsilon = 0.005$, and the learning rate is 0.04.

In linear self-attention and the quadratic network, we set $H = 10$, not 50 as the other architectures, because a large H makes the plateaus very short for these two architectures. This effect was discussed in Section 6 and validated with simulations in Figure 2A.