The Global Empirical NTK: Self-Referential Bias and Dimensionality of Gradient Descent Learning

Anonymous authors
Paper under double-blind review

Abstract

In training a neural network with gradient descent (GD), each iteration induces a linear operator that governs the first-order updates to a model's internal state variables. We define this operator as the Global Empirical Neural Tangent Kernel (NTK_S). In finite-width networks, NTK_S is typically intractable to form, leading prior work to focus on restrictive settings, such as tracking outputs only or taking the limit of infinitely many neurons. Here, we bridge this gap by studying the structure of NTK_S for a wide range of models. Formulating the model state as the solution to a single global implicit constraint, we derive NTK_S as a product of two operators: $\mathcal{K}$, accounting for immediate parameter-to-state interactions, and $\mathcal{P}$, describing internal state-to-state dependencies. For a broad class of weight-based models, including RNNs and transformers, we prove a universal Kronecker-core theorem showing that $\mathcal{K}$ admits an exact, forward-pass-computable form given by the Gram matrix of weight-site variables. This core structure reveals that NTK_S is structurally bottlenecked, constraining its effective rank and giving rise to a *self-referential bias*, whereby GD preferentially learns within dominant modes of joint hidden and input concatenated activity. For recurrent models (GRUs and RNNs), we examine the spectrum of NTK_S and show when it is biased and low-rank in space or time under the proposed decomposition. We further demonstrate that the structure of the model dynamics at initialization biases NTK_S, restricting learning and preventing task components from being learned effectively. Finally, to demonstrate broader applicability, we show that the NTK_S associated with a self-attention transformer is likewise structurally constrained to be low-rank. Overall, we show that NTK_S possesses tractable structure that explains GD bias toward particular task solutions and typical emergence of low-rank representations. To further enable use of NTK_S as a practical metric, we build a library, `kpflow` relying on randomized matrix-free numerical linear algebra.

1 Introduction

Background and Motivation Gradient descent (GD) updates model dynamics indirectly by perturbing its parameters θ. If $h(\theta)$ denotes the model variables of interest, such as the hidden state activities in a neural network, and $J_\theta := D_\theta h(\theta)$ is the corresponding Jacobian with respect to the parameters, then a first-order parameter perturbation $\delta\theta$ induces a change $J_\theta \delta\theta$. Thus, the positive semidefinite operator $J_\theta J_\theta^*$ captures the local geometry through which GD drives corrections in the state space of the variable h. Specifically, it describes the first-order process of transforming task error signals into corrections to h. This operator is the *Neural Tangent Kernel* (NTK), describing at each iteration the local state-space geometry of GD, as well as how the parameters bias first-order training dynamics (Jacot et al., 2018).

Research on the NTK operator has been developed primarily along two complementary directions. One emphasizes infinite-width limits, where the NTK becomes analytically tractable and has been used to study training dynamics and generalization from a variety of perspectives (e.g., Lee et al., 2020; Baratin et al., 2021; Shan and Bordelon, 2022; Canatar et al., 2022; Bordelon et al., 2025). The other perspective studies finite-width *empirical* NTK, often focused on diagnostics, fast computation and approximation of this object, with comparatively fewer works developing explicit analytical characterizations of its structure in broad

contexts (Novak et al., 2022; George, 2021; Mohamadi et al., 2023; Fan and Wang, 2020; Hanin and Nica, 2020; Xiang et al., 2025).

In this work, we formulate and study the *Empirical Global-State NTK* (NTK_S), obtained by collecting all internal variables of a model into a single tensor $h \in S$, which we call the *global state*. For example, in a multilayer perceptron evaluated on a batch of inputs, h contains the activations at every layer and for every input in the batch. For recurrent, implicit, or continuous-time models, h similarly collects all hidden-state trajectories across task inputs. The resulting operator NTK_S therefore acts on the full state space S. This contrasts with many prior studies, which focus on operators that act only on the output space (Jacot et al., 2018). In this sense, the usual output NTK can be viewed as a reduced view of this larger operator. This global formulation is practically useful because it makes the first-order GD dynamics closed at the level of the model state itself, allowing us to study how GD reshapes internal representations and dynamics, not just the output.

Theoretical Findings In practice, NTK_S is intractable to compute explicitly since it contains d^2 when the state has d entries. The hidden state tensor can quickly become prohibitively large, typically scaling with batch size, as well as the layer and hidden unit count. To overcome this, we derive structural theorems demonstrating that the operator is both mathematically tractable and practically predictive. In particular, we show that, for a broad class of models specified by a constraint on all state variables, the NTK_S admits a decomposition into a product of simpler operators, $\mathcal{P}$ and $\mathcal{K}$, describing state-to-state and immediate parameter-to-state gradient propagation, respectively (Proposition 1). In particular, the operator can be written in the form

$$\text{NTK}_S = \mathcal{P}\mathcal{K}\mathcal{P}^*$$

For example, for a recurrent model, $\mathcal{K}$ is an integral operator defining the immediate impact of parameters on the one-step dynamics, while $\mathcal{P}$ is the causal Green's operator describing the model's linearized dynamics (Corollary 1).

Our main theoretical result shows that $\mathcal{K}$, the "core," has an explicit, computable form that can be used to predict the full NTK low-rank spectrum. Specifically, for any model where parameters enter the internal dynamics through matrix-vector products—including models with nesting or recurrence—we show that $\mathcal{K}$ has a universal Kronecker structure (Nielsen and Chuang, 2010),

$$\mathcal{K} = VV^* \otimes I$$

where V collects the activity at the model "weight sites" (Theorem 1). These sites consist of all variables that the model weights multiply. For an RNN, this includes the task input and all hidden activations, while for a self-attention + MLP transformer, these weight sites consist of the task input and attention output (Appendix A.5). Importantly, the weight sites are often both readily computable and of high interest in efforts to interpret model computation and dynamics.

We demonstrate that the core is often a predictive proxy for the spectrum of the full operator NTK_S. In total, NTK_S is decomposable into two interpretable operators: (1) the core VV^* which is architecture dependent and easy to compute, and (2) the operator $\mathcal{P}$, encoding state-space dependencies of the model. Each operator constrains the spectral structure of the NTK_S, manifesting as implicit bias of GD toward particular tasks or low-rank dynamical solutions.

In Corollary 2 we highlight one example of this: *self-referential bias*. Because GD is filtered through the core VV^*, it can only produce corrections lying in the span of the current hidden state and task inputs. This allows us to quantify an important tradeoff: that models producing low-rank dynamics, although often interpretable, can make learning highly restrictive by limiting which tasks are accessible to GD from a given initialization.

Proposition 2 explores how low-rank structure in NTK_S constrains learning. We define reduced views of the NTK that capture its behavior over hidden units (space) and over batches and timesteps (time). Crucially, if either of these reduced operators is low-rank, then GD is biased to produce low-rank state-space corrections. Thus, for GD to produce expressive updates, NTK_S must be sufficiently rich in both its temporal structure

(e.g., capturing both low- and high-frequency features) and its spatial structure, both of which we measure explicitly in Section 5.3.

- Proposition 1 derives the NTK decomposition $\mathrm{NTK}_S = \mathcal{P}\mathcal{K}\mathcal{P}^*$,
- Corollary 1 characterizes the exact structure of $\mathcal{P}$ and $\mathcal{K}$ for recurrent models,
- Theorem 1 demonstrates the exact, computable form $\mathcal{K} = VV^* \otimes I_n$, for any weight-based model,
- Corollary 2 shows this structure gives rise to self-referential bias,
- Proposition 2 provides minimal, testable conditions for low-rank learning under Theorem 1.

Experiments and Library To computationally illustrate and study the structure of NTK_S, we develop a library, `kpflow`, relying on randomized matrix-free numerical linear algebra (Appendix A.1). These approaches lead speedups of multiple orders of magnitude over constructing the full empirical NTK and doing the same analysis (Hazelden, 2025). Our package maps one-to-one to the theory, supporting tensor products, operator composition, partial traced and randomized trace estimation, so that statements such as Theorem 1 can be easily verified, with syntax as in Appendix 16.

We use the theory developed here and `kpflow` library to study low-rank structure in NTK_S, as well as self-referentially biased learning in recurrent models (GRU and RNN) and non-recurrent models (MLP + self-attention transformer). In each case, we find that NTK_S is highly structured, with low-rank structure determined by the joint input-hidden state, V. For GRU, we show that initial hidden state dynamics bias the NTK (Section 5.1), so that, as in Corollary 2, a task can be learned only when the temporal modes of the target are reflected in the hidden-state representation (Section 5.2). Furthermore, in a student-teacher setting (Section 5.3), we show how input dimension and model initialization define distinct low-rank spatial and temporal regimes of NTK_S for a recurrent model. Finally, to demonstrate broader applicability, we apply our methods to the transformer architecture. We find that input rank serves as an upper bound on the rank of NTK_S. Consequently, positional embeddings, augmenting the input to be higher rank, can overcome this bottleneck, better conditioning learning for a wider range of tasks (Section 5.4).

2 Related Work

NTK and Linearized Views of Learning A central line of work studies gradient descent through the Neural Tangent Kernel (NTK), which becomes effectively fixed in the infinite-width limit and yields a linearized description of learning (Jacot et al., 2018; Allen-Zhu et al., 2019; Mei et al., 2018; Bordelon et al., 2020). The infinite-width perspective has been extended beyond feedforward networks, including to vanilla RNNs, neural ODEs, and transformers (Alemohammad et al., 2020; Yang, 2020; Feng and Kolter, 2023). Related work on lazy training, feature learning, finite-width kernel dynamics, and kernel regimes further clarifies when learning remains close to this initialization-dependent linearization and how it departs from it in richer regimes (Chizat et al., 2019; Yang and Hu, 2021; Bordelon and Pehlevan, 2023; Bordelon et al., 2024). In contrast, here we propose to study the finite-width *empirical* NTK acting on the *global state* of the model, i.e. all variables involved in evaluation, rather than an infinite-width output kernel derived for a particular architecture. Our framework yields an explicit factorization of this operator, giving predictions about learning bias and rank bottlenecks related to the formation of the hidden state latent dynamics. Finally, our implicit global formulation of the dynamics (Equation 2) is reminiscent of deep equilibrium models (Bai et al., 2019) (DEQ). But while DEQ uses this form to study equilibrium or effectively infinite-depth models and compute exact gradients, we use it to provide a convenient reformulation of a general explicit model, from which the structural properties of NTK_S follow more naturally.

Learning Dynamics in Recurrent and Dynamical Models A complementary line of theory examines learning in recurrent or continuous-depth models through dynamical-systems and gradient-flow viewpoints. Classical work connected backpropagation to adjoint-state methods and optimal control (LeCun, 1988; Pearlmutter, 1995; Pontryagin et al., 1962). Later work extended these approaches to Neural ODEs and related continuous-depth models (E, 2017; Chen et al., 2018). More recently, exact learning dynamics have been derived in special linear recurrent settings, yielding sharp descriptions of how gradient descent builds

long timescales and low-dimensional structure (Bordelon et al., 2025). Our work is similar in spirit to such dynamical viewpoints, however it targets nonlinear, finite-width models where closed-form training trajectories are generally unavailable. Instead of solving gradient flow explicitly, we derive a general operator decomposition that makes NTK_S structurally interpretable across models.

Dynamical Analyses of Learned Recurrent Representations A large empirical and theoretical literature studies trained recurrent networks through their learned dynamical structure, including fixed points, attractors, low-dimensional manifolds, and dynamical motifs (Sussillo and Barak, 2013; Yang et al., 2019; Farrell et al., 2022; Turner and Barak, 2023; Driscoll et al., 2024; Recanatesi et al., 2021; Schuessler et al., 2020). Related perspectives in simplicity bias and low-rank learning likewise emphasize that GD often favors structured, low-dimensional, or shared solutions (Turner and Barak, 2023; Pellegrino et al., 2023; Farrell et al., 2023; Myrov et al., 2026). The study (Pellegrino et al., 2023) is particularly connected to our work, as it derives bounds for the rank of parameter updates in terms of adjoints and state-space corrections. In contrast to this work, our primary concern here is with state-space correction rank and, more broadly, the spectral structure and bias of state-space corrections, utilizing NTK_S.

In recurrent models, related work also suggests that learning and memory can improve near the edge of chaos, where dynamics remain expressive without becoming strongly unstable (Rajan et al., 2010; Mastrovito et al., 2024). Our work is complementary to this literature. Rather than analyzing learned dynamics only after training, we study the operator governing *state corrections during training*. This suggests a notion of *self-referential bias*: NTK_S has a spectrum that is structurally constrained by the span of the model and inputs activity at each GD iteration, biasing learning aligned with task input and model activations. In this sense, our results connect low-rank and task-dependent learning dynamics to the structure of the training operator itself.

Natural Gradient, Fisher Information, and Kronecker Structure Second-order and natural-gradient methods based on structured curvature approximations, such as K-FAC (Martens and Grosse, 2015; Grosse and Martens, 2016; Martens, 2020), are somewhat related to our work in that they also exploit Kronecker structure in objects tied to the Fisher information or NTK. K-FAC, for example, assumes a layerwise Kronecker factorization of the Fisher information to approximate parameter-space curvature and efficiently precondition gradient descent. In contrast, we study the finite-width empirical NTK on the global state space and show that certain operators possess exact Kronecker structure. Thus, the connection is conceptual rather than algorithmic: in our setting, Kronecker structure is derived exactly, not imposed as an approximation.

3 The Global-State NTK Decomposes into P and K

3.1 Any Model as an Implicit Constraint

The Global State Any parameterized, input-driven model fundamentally consists of a system of variables, constrained by the parameters and task inputs. Throughout, we use the following notation.

$$\begin{aligned} h \in S \quad & \text{The } \textit{Global State Space}, \\ x \in X \quad & \text{The } \textit{Task Input Space}, \\ \theta \in P \quad & \text{The } \textit{Parameter Space}. \end{aligned} \tag{1}$$

We let S and X be generic, reflecting the diversity of models encountered in deep learning and control. Here, global signifies that h and x consist of *all* simultaneous variables involved in the model evaluation, rather than a specific snapshot such as the model output or activations at a particular layer. Tracking the full global state makes learning dynamics closed at the state level, such that the task-dependent error signal acting is the only external driving term in GD.

Any Model as a System of Constraints The state, task inputs and parameters are described by a single model-defining constraint $\mathcal{F} : (S, X, P) \to S$. The space of admissible states in S is implicitly specified by

$$\mathcal{F}(h, x, \theta) = 0. \tag{2}$$

Architectural assumptions impose corresponding algebraic structure on the constraint $\mathcal{F}$. For example, for recurrent architectures, the state Jacobian $D_h\mathcal{F}$ is lower-triangular in time, reflecting causal propagation, as in the following illustration.

3.1.1 Discrete Recurrent Example

Briefly, we illustrate the implicit reformulation for a generic recurrent model, to which we return throughout in order to make each additional structural result concrete. In the discrete case, let $x \in \mathbb{R}^{n_x \times n_t \times n_{in}}$ denote a 3-tensor of n_x distinct task inputs. Here, each distinct task input x_j is a trajectory in $\mathbb{R}^{n_{in}}$ evaluated at n_t time points. Assume a recurrent model driven by these inputs and has sequential dynamics given by

$$h_j(t+1) = f(h_j(t),\, x_j(t+1),\, \theta), \tag{3}$$
$$h_j(0) := 0, \text{ for } j = 1, \ldots, n_x;\; t = 1, \ldots, n_t - 1. \tag{4}$$

Then, this can be reformulated as a global-state constraint. Specifically, let $X = \mathbb{R}^{n_x \times n_t \times n_{in}}, S = \mathbb{R}^{n_x \times n_t \times n_h}$ denote the input and state spaces, respectively. So, $h \in S$ is the stacked 3-tensor of all task-conditioned hidden state evaluations. Finally, the dynamics can be equivalently formulated as

$$\mathcal{F}_{rec}(h, x, \theta) := h - f(T_{\downarrow} h, x, \theta) = 0. \tag{5}$$

where $T_{\downarrow} : S \to S$ is a linear transformation shifting times backwards by one timestep, $T_{\downarrow} = I_{n_x} \otimes U \otimes I_{n_h}$ where U is an n_t by n_t matrix with ones on the lower diagonal.

Continuous Case For a continuous time dynamical system with initial condition 0, evaluated at times $t \in [0, \tau]$, the constraint takes the form $D_t h - f(h, x, \theta) = 0$, where $D_t : S \to S$ is the time-derivative operator. This defines the tangential dynamics of the model along each input-conditioned trajectory. In this case, S is an infinite-dimensional space consisting of bundles of input-driven trajectories in $\mathbb{R}^{n_h}$, but the same structural results naturally apply in this continuous setting.

3.2 Implicit Formulation Yields a Natural Factorization

In this section, we define the global-state NTK (NTK_S), which is an operator modeling how error signals are transformed into state corrections under gradient descent (GD).

A model is trained by choosing a loss to minimize, $L : S \to \mathbb{R}$, e.g., comparing a readout extracted from the global state to a set of desired outputs in a supervised case. Assuming $D_h\mathcal{F}$ is locally invertible (i.e., the dynamics are well-posed), we define two linear operators, $\mathcal{P}, \mathcal{K} : S \to S$ and a global error

$$\mathcal{P} := (D_h\mathcal{F})^{-1}, \quad \mathcal{K} := (D_\theta\mathcal{F})(D_\theta\mathcal{F})^*, \quad \text{err}(h) := \nabla_h L. \tag{6}$$

Where $*$ denotes the Hermitian adjoint (conjugate transpose in the finite case). The error, $\text{err}(h) \in S$ encodes immediate loss gradients. For example, if $L(h) = \frac{1}{2}\mathbb{E}_h[\|h - h^*\|^2]$ then $\text{err}(h) = h - h^*$.

Let (h, θ) be a particular state-parameter pair satisfying Equation 2, for which GD produces a perturbation to the model parameters in the direction of steepest descent in P (choosing learning rate 1 for simplicity),

$$\delta\theta := -\nabla_\theta L(h). \tag{7}$$

According to the chain rule and implicit function theorem, we can then compute $\delta\theta = -((D_\theta\mathcal{F})^* \cdot (D_h\mathcal{F})^{-*})(\text{err}(h))$, and the first order correction to h is $\delta h = ((D_h\mathcal{F})^{-1} \cdot D_\theta\mathcal{F})(\delta\theta)$. Combining these and plugging in Equation 6 we obtain

$$\delta h = -(\mathcal{P} \cdot \mathcal{K} \cdot \mathcal{P}^*)(\text{err}(h)).$$

Hence, $\text{NTK}_S = \mathcal{P}\mathcal{K}\mathcal{P}^*$ is the specific linear operator on the (typically tensor-valued) domain S mapping $\text{err}(h)$ to δh. This is summarized in the following Proposition 1 and Figure 1.

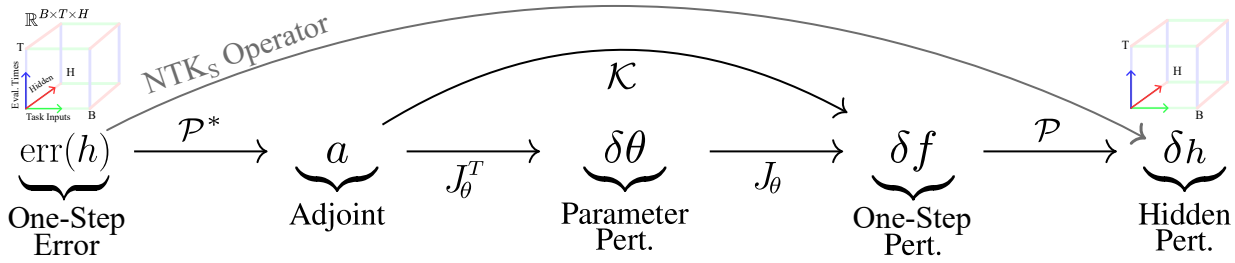

Figure 1: **Backpropagation of errors for a recurrent model, annotated by the corresponding operators, $\mathcal{P}, \mathcal{K}$ and NTK$_S$ from Proposition 1**. The operator $\mathcal{P}^*$ maps the global error signal to state adjoint sensitivity, describing how the loss depends on the state, h. Then, these adjoints are projected into and out of the parameter space by $\mathcal{K}$, potentially zeroing or misdirecting them. Finally, the corrections, which modify the one-step model dynamics, are accumulated over time to produce a correction, δf, by forward propagating through the operator $\mathcal{P}$. We show that, to first-order (small learning rate), backpropagation-through-time and the adjoint method (Rumelhart et al., 1986; Pontryagin et al., 1962; Chen et al., 2018) correspond to applying a sequence of tensor-valued operators acting on the global-state space.

Proposition 1 (NTK$_S$ Definition)**.** *Consider a model in the global implicitly constrained form above,*

$$\mathcal{F}(h, x, \theta) = 0.$$

Each GD step updating θ produces a perturbation to the state, δh, defined by

$$\delta h = -NTK_S(err(h)), \;\; With \;\; NTK_S := \mathcal{P}\mathcal{K}\mathcal{P}^*.$$

Where $\mathcal{P}, \mathcal{K} : S \to S$ are the Propagation and Parameter linear operators, respectively, and $err(h) \in S$ is a global error signal, as defined in Equation 6.

Proof. The proposition follows by the derivation above. In particular, from the global implicit formulation of the model, the factorization emerges naturally from the implicit function theorem and the chain rule. For specific models, however, such as the recurrent example below (Corollary 1), the concrete forms of $\mathcal{P}$ and $\mathcal{K}$ as integral operators can still be nontrivial to derive. □

The decomposition separates three roles: $\mathcal{K}$ aggregates how parameters immediately act on the global state, $\mathcal{P}$ propagates perturbations through the constrained dynamics, and $\mathcal{P}^*$ propagates error signals "backward," as made more explicit in the recurrent example below (Corollary 1) and Figure 1. Importantly, if either of the operators is low-rank, this reduces the rank of the full operator. This, in turn, produces an implicit bias towards a restricted space of corrections. In Theorem 1, we exploit this property, explicitly constructing $\mathcal{K}$ for weight-based models and elucidating how this operator constrains the spectrum of the full NTK.

3.2.1 For Recurrent Models P is the Causal Propagator

We briefly revisit the discrete recurrent example, specializing Proposition 1 to the model in Equation 5. In this setting, the propagation operator $\mathcal{P}$ takes the form of a causal Green's operator, integrating one-step corrections over time, while $\mathcal{K}$ is induced by the one-step parameter Jacobians.

Corollary 1 (Recurrent NTK$_S$)**.** *Consider the discrete recurrent model*

$$\mathcal{F}_{rec}(h, x, \theta) = h - f(T_{\downarrow}h, x, \theta),$$
$$\textit{Where } h \in S := \mathbb{R}^{n_x \times n_t \times n_h}, x \in X := \mathbb{R}^{n_x \times n_t \times n_{in}}, \theta \in P.$$

Then

$$\mathcal{P} = (I - D_h f \cdot T_{\downarrow})^{-1}, \qquad \mathcal{K} = (D_\theta f) \cdot (D_\theta f)^*.$$

Specifically, $\mathcal{P}$ explicitly acts on a collection of tangential perturbations, $\delta f \in S$ by

$$(\mathcal{P}q)_j(t) = \sum_{s \le t} \Phi_j(t, s) \cdot \delta f_j(s).$$

Here, $\Phi_j(t, s) = D_{h_j(s)} h_j(t)$ is the state-transition, describing linear propagation from time s to t, on a fixed task input j. Expanding $NTK_S = \mathcal{P}\mathcal{K}\mathcal{P}^*$ thus yields exact corrections,

$$\delta h_j(t) = -\sum_{t_0=1}^{T} \underbrace{\Phi_j(t, t_0)}_{\substack{\text{Forward} \\ \text{Propagator}}} \sum_{j_1=1}^{n_x} \Big[\sum_{t_1=1}^{T} \underbrace{D_\theta f(h_j(t_0)) D_\theta f(h_{j_1}(t_1))^*}_{\text{Parameter Kernel}} \sum_{t_2=t_1}^{T} \underbrace{\Phi_{j_1}(t_2, t_1)^T}_{\substack{\text{Backward} \\ \text{Propagator}}} \underbrace{err(h_{j_1}(t_2))}_{\text{Error}} \Big].$$

Synopsis For recurrent models, $\mathcal{P}$ is lower-triangular in time, integrating state-space corrections forward through the model dynamics (the causal Green's function). $\mathcal{K}$ captures how parameterization constrains the range of perturbations to the one-step dynamics, f. Finally, $\mathcal{P}^*$ maps the global error signal to an adjoint sensitivity by backpropagation through the Jacobian transpose $(D_h f)^*$. The operator $\mathcal{K}$ then converts this sensitivity into parameter-induced corrections to the dynamics. Finally, $\mathcal{P}$ propagates these corrections forward through the model, yielding the effective state correction δh (Figure 1). Taken together, these operators recover the familiar structure of backpropagation through time and adjoint-based sensitivity propagation in discrete and continuous models, respectively (Werbos, 1990; Chen et al., 2018).

4 The Core K is a Computable Gram Matrix for Weight-Based Models

We now construct $\mathcal{K}$ in Proposition 1 explicitly for weight-based models. Recall that NTK$_S$ is defined as

$$\text{NTK}_S = \mathcal{P}\mathcal{K}\mathcal{P}^*, \qquad \mathcal{P} = (D_h \mathcal{F})^{-1}, \qquad \mathcal{K} = (D_\theta \mathcal{F})(D_\theta \mathcal{F})^*. \tag{8}$$

Notably, a model may be defined by complex, nonlinear state-to-state dependencies, but still have simple immediate parameter-to-state interactions. This observation motivates the theorem below. Specifically, we show that for any weight-based model, $\mathcal{K}$ is a Gram matrix of forward-computable quantities.

Theorem 1 (Universal Kronecker core)**.** *Let a model be defined implicitly by $\mathcal{F}(h, x, \theta) = 0$, where $h \in S$ is the global state. Assume the model is weight-based, in the sense of Appendix A.3.1, and that the associated lifted constraint system is locally well-posed, meaning the model can be evaluated explicitly. Let $V \in \mathbb{R}^{k \times m}$ be the corresponding matrix of weight-sites. Then NTK_S factors as*

$$NTK_S \cong \mathcal{P}_{\text{core}} \big(VV^* \otimes I_n \big) \mathcal{P}_{\text{core}}^*, \tag{9}$$

where $\mathcal{P}_{\text{core}}$ is the induced propagator on S, and $\cong$ denotes equality up to re-ordering of the relevant axes.

Proof. Here, we intuitively outline the three steps of the proof, leaving the full details to the Appendix A.3. The procedure consists of (1) making the parameter structure explicitly by *lifting* the constraint to a higher-

dimensional state space, (2) computing the NTK in this space, (3) projecting back to the original state space S.

(1) Lifted Constraint The weight-based assumption corresponds to a larger system of constraints where the parameter dependence appears only through explicit matrix-vector products on intermediate weight-site variables. Specifically, we introduce a larger global state space, $S_{aug} = (S, \mathbb{R}^{k\times m}, \mathbb{R}^{k\times n})$ and the augmented state

$$h_{\text{aug}} = (h, V, Q), \tag{10}$$

where $V \in \mathbb{R}^{k\times m}$ are the weight sites and $Q = VW^\top$ is the image of V under W. Here k is the number of instances where the weights enter $\mathcal{F}$. This yields an equivalent constraint system,

$$\mathcal{F}_{\text{aug}}(h_{\text{aug}}, x, W) = 0. \tag{11}$$

whose restriction to the original h-coordinates reproduces the original model. This step is necessary to incorporate the simplifying assumption that any weights appear inside the computation as matrix-vector products.

(2) Applying Proposition 1 to the Lifted System Using this lifted system of constraints, we can apply Proposition 1 to compute NTK_S for the lifted state space, S_{aug}. Since the larger system of constraints only reflects the parameter dependence through the matrix-vector products $Q - VW^\top = 0$, the corresponding parameter operator can be written as a 3-by-3 block matrix (in the lifted state space) as

$$\mathcal{K}_{\text{aug}} = (D_W\mathcal{F}_{\text{aug}})(D_W\mathcal{F}_{\text{aug}})^* = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & VV^* \otimes I_n \end{pmatrix}. \tag{12}$$

Next, the operator $\mathcal{P}_{aug} = (D_{h_{aug}}\mathcal{F}_{aug})^{-1}$ is resolved by inverse of a 3-by-3 block matrix.

(3) Project to the Original State The lifted NTK acts on $(S, \mathbb{R}^{k\times m}, \mathbb{R}^{k\times n})$, so the upper left block matrix corresponds to the original NTK_S operator acting on S. Computing this upper block by restriction results in

$$\text{NTK}_S = \mathcal{P}_{\text{core}}(VV^* \otimes I_n)\mathcal{P}^*_{\text{core}}. \tag{13}$$

where $\mathcal{P}_{\text{core}}$ can be resolved exactly; full details are given in Equation 20 in the Appendix. □

Interpretation For simpler notation, throughout the rest of the paper we drop the core prefix. Theorem 1 gives an equivalent decomposition of NTK_S in which the explicit *weight-based* parameter structure makes $\mathcal{K}$ concrete and computable. In particular, the theorem applies to models whose trainable parameters, after lifting if necessary, enter through products of the form Wv for a shared weight matrix $W \in \mathbb{R}^{n\times m}$; we refer to the corresponding internal vectors v as the *weight-sites*. Thus, $\mathcal{P}$ is not a new propagator unrelated to Proposition 1, but the effective state-to-state propagator induced on S by this lifted decomposition. This applies across a broad class of architectures, including nonlinear, recurrent, gated, implicit, and attention-based models, since the theorem depends only on how the weights enter the computation. A formal definition is given in Appendix A.3.1, and concrete instances for the RNN, GRU, and transformer are given in Appendix A.5. The proof procedure–lifting under simplifying assumptions, using Schur complements to compute $\mathcal{P}$ as an inverse, and then projecting back–also provides a general strategy for imposing additional restrictions on the model structure.

The factor VV^* records covariances of activity across the weight-sites, and is therefore directly computable from forward-pass quantities. Moreover, it is readily interpretable: as made explicit in the example below, its principal components correspond to dominant temporal modes across time and task inputs. The spatial Gram matrix V^*V is also closely related to quantities that appear in many prior studies, such as hidden-state or input covariance, and has the same effective rank as VV^*. Thus, systems with low-dimensional structure in their inputs and activity–as, for example, widely observed in neural systems (Recanatesi et al., 2022; Cunningham and Yu, 2014; Farrell et al., 2023; Morales et al., 2023)–will display bottlenecks in the rank of the full NTK, as made explicit in Proposition 2. For many concrete models, including those treated in

Appendix A.5, the weight-sites are straightforward to identify, so the factorization can often be applied directly without explicitly repeating the general lifting and projection argument.

We note again that (Pellegrino et al., 2023) isolate quantities related to those in Theorem 1, and use them to derive elegant bounds on the spectrum or rank of RNN weight updates and, in some cases, RNN state updates, closely related to the bottlenecking ideas explored here. Specifically, (Pellegrino et al., 2023) show how the singular values of updates are bounded by singular values of the adjoint, related to our $\mathcal{P}$ operator, and state covariances, related to our $\mathcal{K}$. Theorem 1 enables analogous concepts to apply to a broader set of models and, as we will show below, invites a separate treatment of temporal and spatial rank concepts with many applications.

The importance of Theorem 1 for the remainder of the paper is that it standardizes NTK_S into two interacting objects: a model-dependent propagator $\mathcal{P}$ and a universal weight-site Gram matrix VV^*. This theorem underlies the findings below. In particular, it lets us distinguish temporal and spatial bottlenecks, derive rank constraints on GD updates, and explain why GD is preferentially steered along modes already present in the current collection of weight-sites, which we call *self-referential bias*.

4.1 Example: Recurrent Model with Input and Hidden Weights

For the discrete recurrent example, consider the discrete-time system $h_{t+1} = f(h_t,\ W_{\mathrm{rec}}h_t,\ W_{\mathrm{in}}x_{t+1})$, with parameters $\theta = \mathrm{cat}(\mathrm{vec}(W_{\mathrm{rec}}), \mathrm{vec}(W_{\mathrm{in}}))$, where cat forms the direct sum concatenation. Collecting the weights into the block matrix $W = \mathrm{blockdiag}(W_{\mathrm{rec}}, W_{\mathrm{in}})$, the model depends on W only through the concatenated state-input vectors $v_t = \mathrm{cat}(h_t, x_{t+1})$ (Figure 2). These are precisely the weight-sites, so the corresponding matrix is

$$V = \mathrm{cat}(H, X) \in \mathbb{R}^{n_x n_t \times (n_{\mathrm{h}} + n_{\mathrm{in}})}.$$

Thus, in the notation of Theorem 1, we have $k = n_x n_t$, $m = n_{\mathrm{h}} + n_{\mathrm{in}}$, and $n = 2n_{\mathrm{h}}$. Theorem 1 therefore yields

$$\mathrm{NTK}_S = \mathcal{P}\big(VV^* \otimes I_{2n_{\mathrm{h}}}\big)\mathcal{P}^*. \tag{14}$$

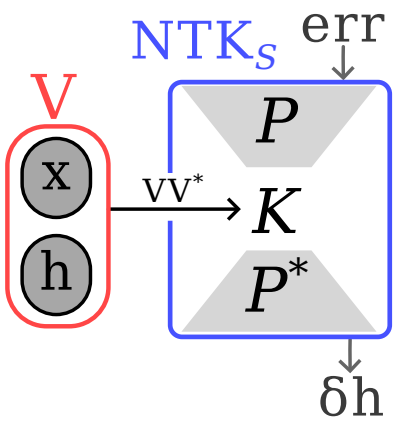

Figure 2: Schematic of Theorem 1 for the Example 4.1, with $V = \mathrm{cat}(H, X)$, as is the case for the GRU and RNN. If the hidden units and inputs have low-dimensional activity, then this joint state matrix bottlenecks the full NTK.

Here, the Gram matrix has explicit structure, composed of all contracted inner products of the hidden and input states. Specifically, we can write

$$(VV^*)_{j,t;j',t'} = \langle h_j(t), h_{j'}(t')\rangle + \langle x_j(t), x_{j'}(t')\rangle$$

Explicitly, it is an uncentered covariance matrix corresponding to the joint hidden-input activity. Thus, the matrix VV^* acts like an (unnormalized) projector onto the dominant modes of the joint hidden-input activity over all timesteps and batches. Low-rank structure in $\mathrm{cat}(h, x)$ directly bottlenecks the accessible NTK corrections, giving rise to low-rank, biased learning, as formalized in the next section.

Freezing Parameters Finally, note that $\mathcal{K}$ captures the choice of dynamical parameters and $\mathcal{P}$ the linearized model dynamics. Specifically, if we choose to fix the hidden weights, the core is reduced to the form $VV^* = XX^*$, reflecting that only the input weights are trained. Likewise if we fix the input weights, the core becomes $VV^* = HH^*$, the hidden state Gram matrix (motivating Figure 5). In either case, however, $\mathcal{P}$ does not change. On the other hand, if a model has the same mechanism for parameter entry but different linearized dynamics, the computational form of $\mathcal{K}$ remains identical while $\mathcal{P}$ changes. This shows that $\mathcal{P}$ and $\mathcal{K}$ can be modified effectively independently of one another, with very different roles.

4.2 GD is Self-Referentially Biased Towards Core Modes in V

Theorem 1 shows that the NTK_S is mediated by the current weight-site state V. Thus, gradient descent is not equally responsive to all error directions: it learns most effectively through modes represented in the

model's current weight-site feature matrix. We refer to this mechanism as **self-referential bias**, where the model's current activity pattern determines which corrections are accessible to the optimizer.

The corollary below shows that $\langle \mathrm{NTK}_S \mathrm{err}(h), \mathrm{err}(h) \rangle_F$, the rate at which the loss decreases under one GD step, depends on how $\mathrm{adj}(h)$, defined below, aligns with V, over all time and trials. When $\mathcal{P}$ is roughly isotropic (such as with low recurrence), this reduces to direct overlap between the error and the weight-sites. By contrast, when $\mathcal{P}$ is anisotropic (such as when there are exploding gradients in a single dominant direction), it compounds with $\mathcal{K}$ to further reduce the rank of NTK_S and effective bias of learning, as explored in Section 5.3. This implies that the alignment between the task target (or individual SVD modes of the target) and the core, V, can be a useful and easy to compute proxy for how much the NTK aligns with the particular task. This relationship is explored computationally in Experiment 5.2.

Corollary 2 (Self-referential bias via adjoint alignment)**.** *Let $\mathcal{P}$, $\mathcal{K}$, and err be as in Proposition 1, and define the global adjoint state by*

$$adj(h) := \mathcal{P}^*(err(h)).$$

If, moreover, $\mathcal{K} = VV^* \otimes I_n$ as in Theorem 1, then

$$\langle NTK_S\, err(h), err(h) \rangle_F = \|(V^* \otimes I_n) adj(h)\|_F^2.$$

Thus, at each iteration, V acts as a filter, removing any components of $\mathrm{adj}(h)$ *orthogonal to V, so that GD it restricted to corrections aligned with the weight-sites.*

Proof. This is a direct consequence of Theorem 1. Specifically, the NTK alignment expands as

$$\langle \mathrm{NTK}_S \mathrm{err}(h), \mathrm{err}(h) \rangle_F = \langle \mathcal{P}(VV^* \otimes I_n)\mathcal{P}^*(\mathrm{err}(h)), \mathrm{err}(h) \rangle_F .$$

We can pass $\mathcal{P}(V \otimes I_n)$ to the right side of the inner product by taking its adjoint, yielding

$$= \langle (V^* \otimes I_n)\mathcal{P}^*(\mathrm{err}(h)), (V^* \otimes I_n)\mathcal{P}^*(\mathrm{err}(h)) \rangle_F .$$

Finally, from the definition of $\mathrm{adj}(h)$, this equals $\|(V^* \otimes I_n)\mathrm{adj}(h)\|_F^2$. □

Remarks Corollary 2 gives an operational form of self-referential bias. The adjoint operator $\mathcal{P}^*$ transports the error signal into state-space, producing the global adjoint state $\mathrm{adj}(h)$. Then, the Kronecker core measures the overlaps of this state with the span of the current weight-sites. The geometry of V already determines which error directions can effectively drive learning. If $\mathrm{adj}(h)$ has little overlap with the dominant modes of V, learning stalls, corresponding to a poorly conditioned NTK_S for the task.

In the discrete recurrent example above, $V = \mathrm{cat}(h, x)$. This means that if the joint hidden-input state is concentrated in a few dominant modes over time, trials, or hidden units, then GD is correspondingly biased to learn through those same modes. For example, if V corresponds to low frequency dynamics over all time and trials, NTK_S effectively cancels any high-frequency components of the target. If the activity of V drops off in a particular time period (e.g., in the response period in the Memory-Pro task below), then parts of the target in this range are not learned. Further, if V corresponds to low-rank latent dynamics in space, meaning that it can be projected to a low-dimensional PCA space, then corrections are constrained by the low-rank Gram matrix VV^T, as we study in the next section. As a result, we define this as the basic mechanism of self-referential bias, potentially explaining more specific observed phenomena, including inductive bias of GD towards low-frequency parts of a task, or neural collapse, biasing GD towards low-dimensional latent dynamics (Rahaman et al., 2019; Cao et al., 2021; Farrell et al., 2022; Recanatesi et al., 2021; Farrell et al., 2023; Freedman and Assad, 2006).

4.3 Necessary Requirements for Low-Rank Learning (Over Space and Time)

We have refrained from vectorizing the global-state tensor h throughout. Here, we show that, in conjunction with Theorem 1, this allows us to define multiple reduced views of the NTK operator, capturing its dominant

behavior over hidden units (space) or batches and timesteps (time). The core $VV^* \otimes I_n$ from Theorem 1 acts non-identically only on the temporal part of this view. Importantly, Proposition 2 shows that if *either* the reduced spatial or temporal view of the NTK is low-rank, then the updates δh it produces are low-rank in *both* axes. Thus, high-rank learning necessitates that both the spatial and temporal ranks of the NTK are high. We will undertake a concrete application of this in Section 5.3 below, where we explore distinct regimes in which either the spatial or temporal rank of the NTK is the limiting factor.

Space-Time Reduced Views of the Global-State NTK Theorem 1 shows that *any* weight-based model exhibits a decomposition of the NTK_S over the tensor product domain $\mathbb{R}^k \otimes \mathbb{R}^n$. Here, k indexes the vectorized batch-time axis, $k = n_x \cdot n_t$, while n is the physical neuron dimension. Using this as motivation, we call the left factor of such a tensor product the *temporal part* and the right factor the *spatial part*. Specifically, we define two reduced operators,

$$\text{NTK}_S^{temp} := \frac{\text{tr}_n(\text{NTK}_S)}{n} \in \mathbb{R}^{k \times k}; \quad \text{NTK}_S^{space} := \frac{\text{tr}_k(\text{NTK}_S)}{k} \in \mathbb{R}^{n \times n}. \tag{15}$$

where tr_n denotes the *partial trace*, effectively summing along one axis of the operator. Note that NTK_S is positive semidefinite, so this corresponds to a reduced second-moment view of the original global-state parameter sensitivity. In Appendix A.1, we detail how these operators can be implemented efficiently in a matrix-free way. These views of the NTK apply to any weight-based model covered by Theorem 1. Indeed, we compute these ranks for a non-recurrent self-attention transformer in Section 5.4 below.

Illustration We first give a simple demonstration of the relationship between low-rank spatial and temporal properties via the operator $\mathcal{K}$, which is always Kronecker separable by Theorem 1. Specifically consider a one-dimensional case where a single temporal mode dominates, $\mathcal{K} = vv^T \otimes I_n$, where $v \in \mathbb{R}^k$ is a vector, representing the same pattern of activity shared across all the joint hidden-input activity. Then, the operator is rank one temporally, but full rank spatially. Let $\text{err}(h) \in \mathbb{R}^{k \times n}$. Define $q := v^T \text{err}(h) \in \mathbb{R}^n$. Then,

$$(vv^T \otimes I_n)(\text{err}(h)) = vq^T \in \mathbb{R}^{k \times n},$$

so every row of the output is a scalar multiple of q. Thus, all updates lie in a one-dimensional spatial subspace, exploring only a single direction in $\mathbb{R}^n$. In other words, the low-rank temporal structure of $vv^T \otimes I_n$ implies a spatial bottleneck on every output of this operator.

Next, we generalize and formalize this phenomenon in Proposition 2.

Proposition 2 (Space-Time *NTK* Bottleneck)**.** *In the context of Theorem 1, NTK_S is an operator acting on $\mathbb{R}^{k \times n}$, and we can define its reduced temporal and spatial views (Equation 15). Let $err(h) \in \mathbb{R}^{k \times n}$ denote any input error signal, and let $\delta h = NTK_S(err(h))$ be the induced correction. Then,*

$$rank(\delta h) \leq \min\big(rank(NTK_S^{space}), rank(NTK_S^{temp})\big),$$

where $rank(\delta h)$ is the matrix rank of $\delta h \in \mathbb{R}^{k \times n}$, bounding both its spatial and temporal expressiveness.

Proof. See Appendix A.4 for a complete proof, relying primarily on the fact that the NTK_S, being a Gram operator, is positive semidefinite. In essence, we show that

$$u^T \delta h = 0 \in \mathbb{R}^n \quad \forall u \in \ker(\text{NTK}_S^{temp}).$$

In turn, this implies that the range of corrections the NTK can produce must lie within the temporal image of NTK_S^{temp}. Since column rank and row rank of a matrix, e.g. of a correction $\delta h \in \mathbb{R}^{k \times n}$, are identical, this yields the rank bound from NTK_S^{temp}. Similar reasoning yields the bound from NTK_S^{space}. □

Remarks This proposition shows that biased low-rank learning emerges when *either* of the two partially reduced operators has low-rank. A low-rank NTK at a particular iteration of GD implies that learning may be

highly biased toward particular targets or low-complexity updates, such as failing to capture high-frequency temporal structure in the target of a recurrent task. In the special case where $\mathcal{P}$ is explicitly separable across space and time, meaning that propagation can be written as a tensor product of independent spatial and temporal operators, this yields the exact bound

$$\operatorname{rank}(\delta h) \leq \operatorname{rank}(VV^*)\,.$$

showing that the spatial and temporal complexity of updates are exactly bottlenecked by the Kronecker core VV^*. Even outside of this case, the spatial and temporal partial views, along with their associated singular values and ranks, can be computed efficiently in our framework (Appendix A.1). Finally, we remark that high-rank latent representations may still emerge over the course of GD, even when updates are low-rank at every iteration. Only under additional simplifying assumptions that allow the NTK to remain controlled over training (e.g., lazy regimes or linear models) can such a conclusion be made, since it requires tracking the full course of latent formation across all GD iterations (see Future Directions).

5 The Kronecker Core Predicts Learning Bottlenecks Across Architectures

5.1 GRU Joint Hidden-Input Activity Predicts Dominant NTK Modes

GRU Global-State Core and NTK For the GRU with n_h neurons, the core of the NTK is $VV^T \otimes I_{3n}$, where V consists of the concatenated hidden-state and input activity,

$$V = \operatorname{cat}(h, x) \in \mathbb{R}^{n_x \times n_t \times (n_h + n_{in})}\,,$$

and $3n$ comes from the three-fold repetition of weight-sites in this model (derivation in Appendix A.5). Figure 3A illustrates the Memory-Pro, in which a two-dimensional stimulus is provided to the model during an initial stimulus period, it must retain knowledge of this stimulus during a memory period and finally should respond with the identical stimulus provided during a final, response period. Here, the stimulus has the form $\cos(\theta), \sin(\theta)$ for uniform angle $\theta \in [0, 2\pi]$, so the latent dynamics the model forms on this task often resembles an attractor to a two-dimensional ring (Driscoll et al., 2024). Finally, the model is provided with a fixation input, which is set to 1 during the stimulus and memory periods and switches to zero during the response period, indicating that the model should respond.

Findings Figure 3B measures the alignment between the NTK_S operator and its core, $VV^* \otimes I_{3n}$, over multiple iterations of GD on the Memory-Pro task. It shows that, throughout training on this task, the core provides a good predictor of the spectrum of the full NTK, meaning that it captures the dominant modes of the NTK. This point is further reinforced by Figure 3C, which compares the spectrum of NTK_S to those of the operators VV^* and $\mathcal{P}$ from which it is composed, at the outset of GD training. Specifically, we measure the dominant *temporal modes* of the operators involved, as defined in Section 4.3. These modes can be viewed as matrices in $\mathbb{R}^{n_x \times n_t}$, describing, for each distinct task input, the dominant temporal structures to which the operator responds. Using this perspective, we observe that $\mathcal{P}$ is effectively full-rank, with singular values that are relatively evenly distributed across the range of outputs it can produce. In contrast, the core VV^* is low-rank (with only 3 modes needed to capture 95% of its variance). Intuitively, this suggests that $\mathcal{P}$ is roughly isotropic and does not strongly bias the direction of corrections. On the other hand, the Kronecker core VV^* strongly biases corrections, acting highly anisotropically. Finally, we find that NTK_S is itself low-rank, with its cumulative variance curve lying strictly above that of VV^* and modes well predicted by this Kronecker core matrix.

Figure 3D shows that the leading temporal modes of the NTK are highly structured, concentrating on a small family of response profiles across the trial. These modes are not generic basis functions; they are induced by the task-dependent activity present at initialization. Consequently, even before training, gradient descent is constrained to act most strongly along a narrow subspace of temporal directions selected by the current recurrent activity.

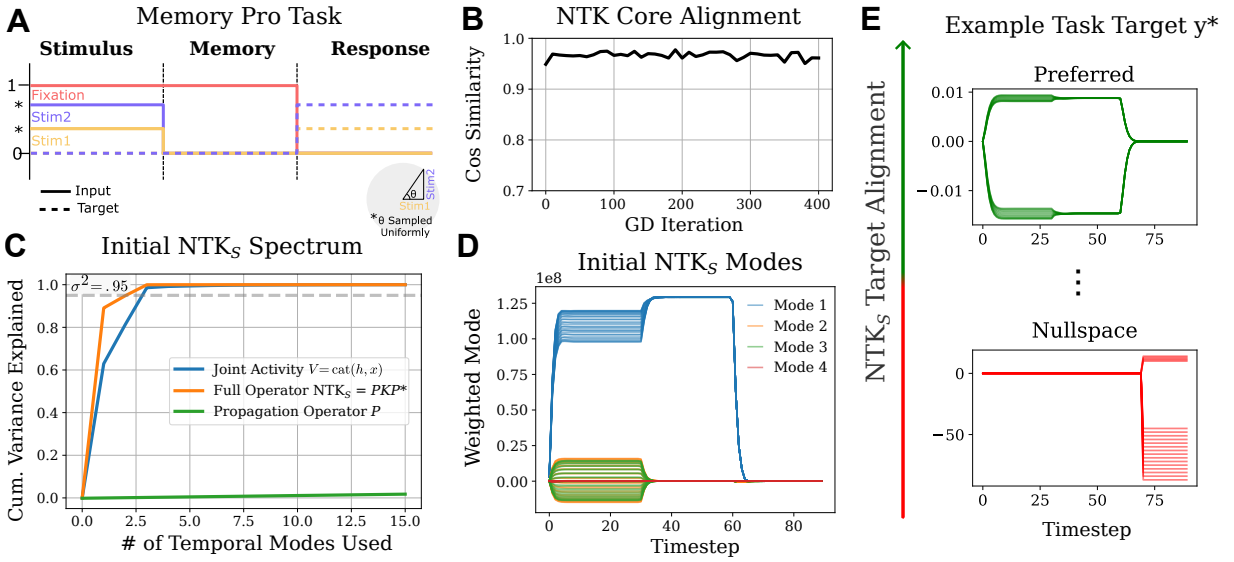

Figure 3: **Temporal bottlenecking of the global-state NTK by the Kronecker core VV^T. A** Schematic of the Memory-Pro task, in which the model must reproduce a two-dimensional stimulus after a delay period. **B** Cosine similarity between the core, $\mathcal{K} = VV^* \otimes I_n$, and the NTK, $\cos(\text{NTK}_S, VV^T \otimes I_n)$, over GD training of the GRU model on the task in A, showing that the two operators share a similar common basis. Here, we use a model with Xavier initialization, corresponding to Network 1 in Figure 4 below. **C** Cumulative variance explained by the temporal spectrum of the full NTK, the Kronecker core VV^T, and the propagation operator $\mathcal{P}$. The full NTK has effective rank close to that of VV^T, while $\mathcal{P}$ remains comparatively high-rank in this regime. **D** Dominant temporal modes of the NTK. Each individual mode is a matrix $\mathbb{R}^{n_x \times n_t}$, describing the time and trial structure of a particular input to the operator (formally defined in Section 4.3). Thus, a single mode consists of $n_x = 20$ here, as shown by the multiple curves of each color in the plot. Importantly, the NTK modes encode the structure of the input and hidden state, dropping to zero in the response period (timesteps 60-90 here). **E** Example one-dimensional task targets in $\mathbb{R}^{n_x \times n_t \times 1}$, over batches and time. The top is an example of a task that would highly align with the NTK, i.e., its activity is well-explained by the dominant modes in D. By contrast, the second target is badly aligned, since it primarily resides in the range where the NTK modes are zero. Interestingly, the Memory-Pro task has a target similar to the bottom row since it requires a delayed response. Section 5.2 explores the consequences of this for learning, showing that this particular GRU is unable to learn the full Memory-Pro task with SGD.

5.2 Self-Referential Bias Stalls GD When Target Modes are Absent in The Initial NTK

Self-Referential Bias As suggested by the bottom row of Figure 3E, some task targets can have weak alignment with the dominant temporal modes of NTK_S at initialization, meaning that their projection onto the high-eigenvalue eigenspaces is small. In this subsection, we examine this in detail and show that such NTK-target misalignment explains the self-referential bias predicted by Corollary 2: because GD corrections are constrained by the current hidden and input activity, learning is initially biased toward target components already represented in the model dynamics, while poorly aligned components receive only very small updates and therefore learn slowly, producing plateaus.

Two GRU Initializations To study this, we contrast two GRUs trained on the Memory-Pro task. Network 1 is the default initialization used in the previous subsection. At initialization, when the input is removed during the response period, its hidden dynamics collapse stably to the trivial fixed point at zero (Figure 4A, pre-training). Network 2 is initialized so that, under zero input, the model instead exhibits five non-trivial stable fixed points (dynamics on the task in Figure 4B pre-training; construction in Appendix A.2.2). Thus the two models begin with qualitatively different latent dynamics, and consequently with different weight-site activity $V = \text{cat}(h, x)$, even before training.

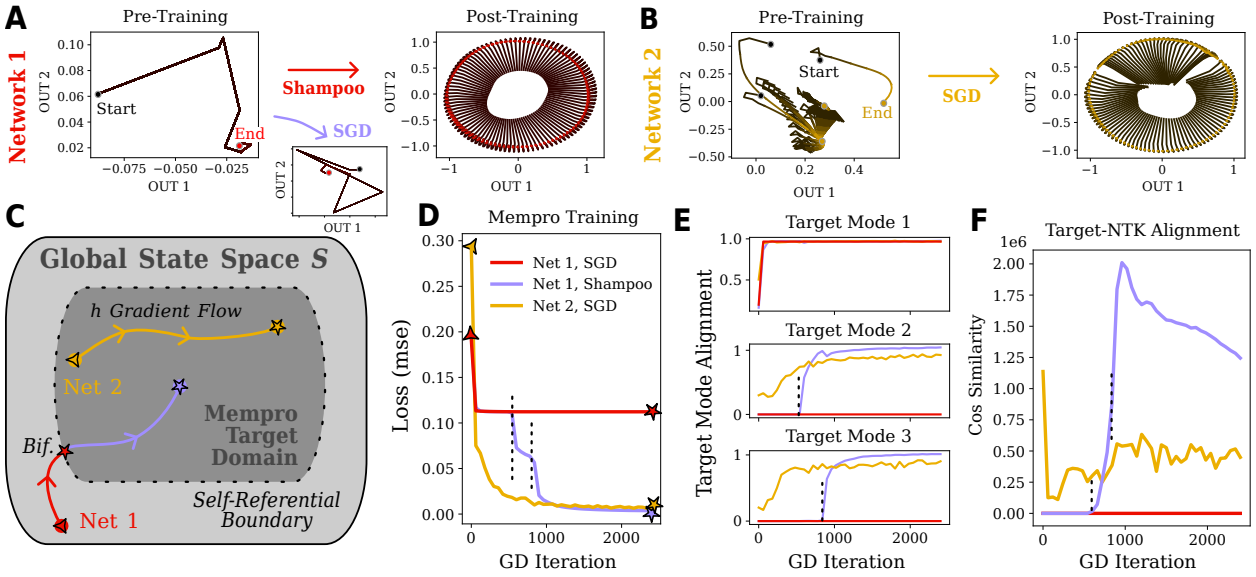

Figure 4: **Self-referential bias can stall SGD on the Memory-Pro task. A-B** Outputs before and after training for two GRU initializations. **A** (Network 1): a default Xavier initialization with weight scale 1, whose hidden dynamics collapse to a single fixed point under zero input during the response period, regardless of the input ("End" in left panel of Figure). **B** (Network 2): as an illustrative case, an initialization chosen to produce 5 non-trivial fixed points under zero input (Appendix A.2.1), showing that the task-driven dynamics have multiple endpoints at the final evaluation timestep. As in further panels, this initialization leads to better NTK-target alignment initially. Thus, the two networks begin with qualitatively different latent dynamics before training. **C** Schematic of the corresponding gradient flows in global state space S. Network 2 begins in a regime already containing the qualitative structure needed for the task, whereas Network 1 must first create this structure during training. **D** Training losses for the two initializations. Network 2 trains successfully with SGD, while Network 1 plateaus under SGD and only solves the task when trained with the second order Shampoo optimizer (Gupta et al., 2018) In the latter case, a transition occurs around iteration 750, where new non-trivial hidden-state structure emerges. **E** Alignment of the task target with its leading three temporal modes during training. Network 2 begins with non-trivial alignment in all three modes and learns them steadily with SGD. By contrast, Network 1 under SGD learns only the first mode, while the higher modes remain effectively absent. Training Network 1 with Shampoo eventually produces these missing modes. **F** Cosine alignment of the NTK with target mode 3 during training. For Network 1, the alignment is initially nearly zero, consistent with the near-null target modes identified in Figure 3E. Under Shampoo, NTK-target alignment rises before the corresponding hidden-state mode becomes clearly visible in **E**, consistent with the operator first becoming aligned with the missing target mode and then driving it into the dynamics.

Throughout this subsection, we measure alignment with the target modes obtained from SVD on the target matrix. Specifically, the target has shape $n_x \cdot n_t \times 3$, due to the three outputs, so its left singular vectors define three orthogonal modes over time and batches, $U \in \mathbb{R}^{n_x \cdot n_t \times 3}$.

Network 1 Fails to Learn Some Target Modes The distinction in initial latent dynamics between the two networks is important because, since the dominant temporal modes of the NTK are already largely determined by the activity-induced core VV^T when $\mathcal{P}$ is well-conditioned, as shown in the previous subsection. For the Memory-Pro task, the target lies mainly in the response period, whereas the external stimulus is only present earlier in the trial. Hence, if the hidden state collapses during the response period, then the target modes have almost no overlap with the temporal modes induced by V. This is exactly what happens in Network 1: both the input and hidden activity are nearly absent during the response period. Thus, the target is initially very poorly represented in the Kronecker core, VV^T, hence not well represented in the spectrum of NTK_S. On the other hand, Network 2 maintains non-trivial hidden activity during this period, with V

overlapping with all three target modes and substantially. This in turn improves NTK-target alignment from the outset.

Training dynamics in Figure 4D-F reflect the distinction. Network 2 learns the task steadily with SGD, while Network 1 quickly reaches a plateau. Figure 4E shows the reason. In all cases the first target mode is learned, but for Network 1 under SGD the higher target modes remain essentially absent. In other words, SGD can only exploit the modes already present in the initial NTK_S geometry. This is the self-referential bias of learning in a concrete form: the model first learns what its current dynamics already support.

Figure 4F makes this still more explicit and tracks the minimum NTK-target alignment throughout training. For Network 1, NTK-target alignment is effectively zero, consistent with the near-null target modes identified in Figure 3E. As a result, the corresponding corrections produced by SGD are extremely small, and training stalls, plateauing after the first mode is learned. In contrast, Network 2 begins with substantially higher target alignment for all three target modes, and is therefore trainable by SGD alone.

Resolving Poor NTK Alignment with Curvature-Aware Optimization To test whether this plateau is really caused by poor NTK-target alignment, rather than by a generic limitation such as insufficient training time, poor learning-rate tuning, or an inaccessible parameter-space solution, we also train Network 1 with Shampoo (Gupta et al., 2018). Shampoo adaptively rescales parameter updates using an approximate second-order preconditioner. In this case, where the hidden state is low-rank and biased towards certain temporal modes, the preconditioner effectively aims to recondition the spectrum of the Gram core matrix HH^T. Thus, directions that are nearly inactive under the original GD kernel can receive larger effective updates. In this case, the optimizer is able to amplify nearly-null directions enough to escape the plateau. Around iteration 750, a new non-trivial mode emerges in the hidden dynamics, after which the remaining target modes become learnable (Figure 4D-F). Notably, the NTK-target alignment begins to rise before this new activity is clearly visible in the hidden dynamics, suggesting that the operator first becomes aligned with the missing target mode and then drives it into the latent state.

This experiment illustrates one concrete consequence of self-referential bias. When the initial hidden dynamics already contain activity aligned with the task, the corresponding core VV^T and NTK are well conditioned for learning, and SGD succeeds. When these modes are absent, the relevant target directions are effectively null under the NTK, producing a long plateau or complete failure of iterative learning under SGD. Thus, there is a **trade-off** between structured low-rank dynamics making models stable and interpretable (Farrell et al., 2023; Mastrogiuseppe and Ostojic, 2018; Ostojic and Fusi, 2024), but also biasing learning toward those same directions, restricting the range of tasks that SGD can learn efficiently from a given initialization. Intriguingly, this effect may explain the fact that fixed points in model activity are reused across similar tasks Driscoll et al. (2024), since NTK_S is implicitly constrained by the dynamics only allowing for a highly reduced range of corrections that it can produce.

5.3 Low-Rank Space-Time Learning Regimes Based on Rank of P and K

We further examine the low-rank structure of NTK_S, to isolate distinct regimes in which gradient descent is bottlenecked in its rank over space, time or both. Specifically, we compute the effective rank of the partially reduced views, $\text{NTK}_S^{temp} \in \mathbb{R}^{n_x \cdot n_t \times n_x \cdot n_t}$ and $\text{NTK}_S^{space} \in \mathbb{R}^{n_h \times n_h}$, defined in Section 4.3, for a vanilla RNN with global state space $S = \mathbb{R}^{n_x \times n_t \times n_h}$, over a batch of n_x inputs, simulated for n_t timesteps, with n_h hidden units.

The spectrum of NTK_S^{space} describes the global spatial bias of the operator. If it is concentrated in a few directions, then regardless of the error signal, updates are confined to a low-dimensional subspace of $\mathbb{R}^{n_h}$. In contrast, the spectrum of NTK_S^{temp} describes the range of temporal modes available for solving tasks with a given set of inputs. When NTK_S^{temp} is low-rank, updates are restricted to a small family of temporal signals, causing bottlenecking of spatial corrections induced by a given error (Proposition 2). Thus, the spatial effective rank captures biases introduced by the propagation geometry of the architecture, while the temporal effective rank captures biases introduced by the activity and input patterns available during learning.

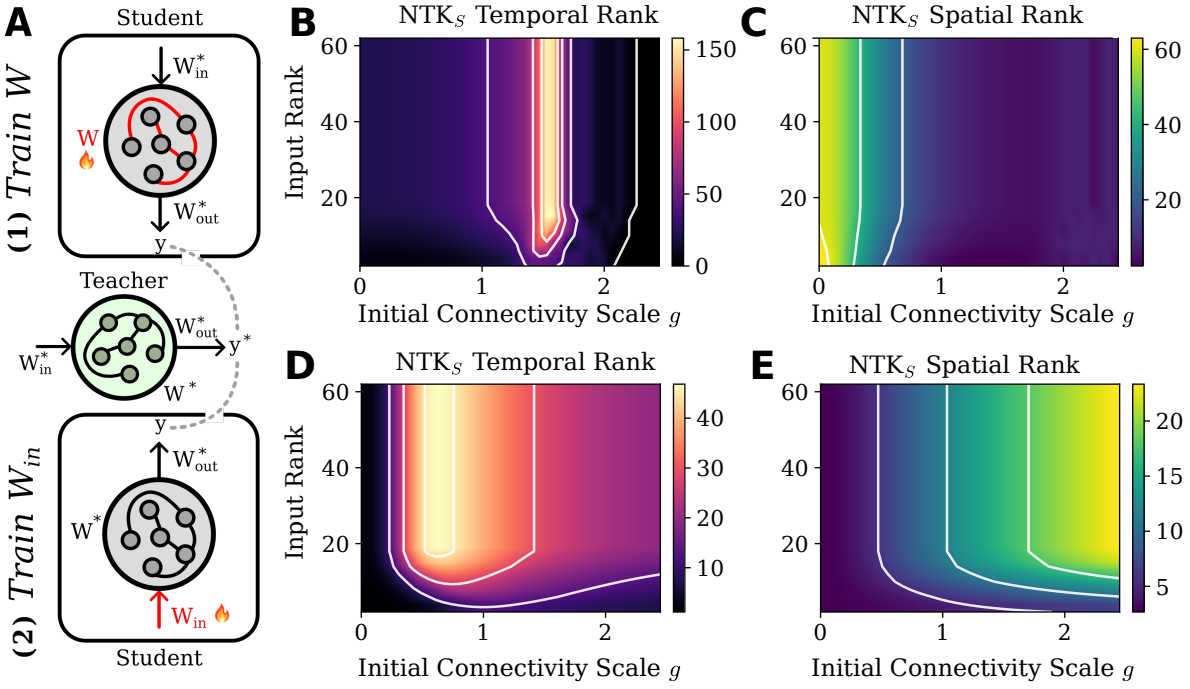

Figure 5: **Recurrent gain and input rank induce distinct spatial and temporal NTK bottlenecks.** **A** Two variants of a student-teacher task. For both, we begin with a vanilla RNN teacher with fixed weights W^*, W^*_{in}, W^*_{out}, all drawn from Xavier normal initialization. In the top task, we freeze the student to have identical weights other than W, which is initialized randomly with gain g and trained with GD. The training objective is that the outputs of the two models on random normal inputs match. In the bottom task, we instead fix $W = W^*$ and randomly initialize W_{in} with gain g, with the same objective. Thus, these tasks isolate the role of learning recurrent versus input weights in GD, with the NTK core defined by hidden or input state, respectively (see example in Section 4.1). The inputs to each model are random normal (see Appendix A.2.2), with an input channel count varied in B–F. **B** Temporal rank when training the recurrent weights. This rank increases with gain g before collapsing again in the unstable regime, consistent with the core prediction that richer recurrent activity increases the temporal rank available to the NTK. **C** Spatial rank when training the recurrent weights. This rank, determined primarily by the propagation operator $\mathcal{P}$, collapses as g increases, corresponding to $\mathcal{P}$ concentrating onto a small number of unstable or exploding modes. **D-E** Temporal and spatial rank when training the input weights. Here, the temporal rank grows with input rank, consistent with the bottlenecking effect predicted by Theorem 1, while the spatial rank varies comparatively little.

In particular, Figure 5 studies these reduced operators in a student-teacher RNN setting, summarized in panel A. We consider two complementary tasks. (1) We set all student weights identically to the teacher other than the recurrent weights, which must be trained with GD so the student and teacher outputs match. (2) Instead, the task requires training of only the input weights, with all other student weights identical to the teacher weights and frozen during training. In both tasks, we investigate initializing the dynamic weights with a random normal distribution and gain g. In addition to sweeping the gain term, we also sweep the effective rank of the task inputs. These two sweeps are chosen to probe the two factors in the decomposition, $\text{NTK}_S = \mathcal{P}(\text{cat}(H, X)\text{cat}(H, X)^T \otimes I_n)\mathcal{P}^*$, demonstrating that the core $\text{cat}(H, X)\text{cat}(H, X)^T$ primarily bottlenecks temporal structure, while the propagation operator $\mathcal{P}$ reshapes the spatial structure.

Temporal Bottlenecks from the Core Figure 5B shows that when training the recurrent weights, the temporal rank of the NTK increases as the recurrent gain g moves away from a strongly contractive regime, consistent with richer hidden activity producing a higher-rank core VV^T and hence a more expressive temporal NTK (Sompolinsky et al., 1988; Rajan and Abbott, 2006; Engelken et al., 2020). However, this

increase is confined to an intermediate regime: for larger g, the temporal rank drops sharply, indicating that the temporal richness induced by the core is ultimately lost once propagation becomes sufficiently ill-conditioned. Thus, increasing gain can initially enrich the temporal geometry available to learning, but beyond a critical regime instability produces a new low-rank bottleneck.

The same mechanism appears when training the input weights. Figure 5D shows that the temporal rank grows strongly with input rank, as predicted by the core decomposition, since the temporal bottleneck is controlled primarily by the rank of the weight-site activity $V = \mathrm{cat}(H, X)$. The dependence on recurrent gain is present but secondary compared with the effect of input rank.

Spatial Bottlenecks from Propagation The spatial rank behaves differently. Figure 5C shows that when training the recurrent weights, the spatial rank collapses as g increases. This collapse is not explained by the Kronecker core, which is separable and isotropic over $\mathbb{R}^{n_h}$. Rather, it is governed primarily by the operator $\mathcal{P}$, which becomes increasingly ill-conditioned as the dynamics approach instability. In this regime, propagation concentrates onto a small number of expanding directions, so that the NTK becomes effectively low-rank in space even when the temporal core remains expressive.

By contrast, Figure 5E shows that the spatial rank varies more gradually, increasing somewhat with input rank but without the sharp transitions seen in the temporal rank. This reinforces the distinction above: input complexity primarily controls the temporal bottleneck through VV^T, while recurrent gain primarily controls the stronger spatial bottleneck through $\mathcal{P}$.

Taken together, these sweeps reveal two distinct bottlenecks in the empirical NTK. The weight-site core VV^T primarily limits the temporal diversity of available corrections, while the propagation operator $\mathcal{P}$ primarily limits how broadly those corrections are distributed across state space. From this perspective, increasing recurrent gain can enrich temporal expressivity, but only until propagation becomes sufficiently ill-conditioned that updates concentrate onto a small number of directions (Sompolinsky et al., 1988; Rajan and Abbott, 2006; Engelken et al., 2020). Favorable regimes for learning a diverse set of tasks therefore require a core rich enough to represent task-relevant corrections and a propagator $\mathcal{P}$ that remains sufficiently well-conditioned to distribute them broadly. In a broad sense, this echoes the influential findings of (Sussillo and Abbott, 2009), regarding optimal learning regarding at the "edge of chaos."

5.4 The NTK Core Bottleneck Appears in a Self-Attention Transformer

Our analysis is not confined to recurrent models, although they are the primary focus of this work. Indeed, Theorem 1 relates to any model that can be written as a system of constraints relating the state variables, inputs and weight-based parameters. This encompasses a wide range of models, including any explicit, weight-based model. To illustrate this structural theorem in a distinct setting, we apply the same perspective to a nonlinear self-attention block of a transformer. We use this example to show that the same Kronecker-core mechanism appears in a non-recurrent model as well: low-dimensional input structure can bottleneck the *temporal* rank of the NTK, while widening the attention block primarily affects its spatial and overall rank. We note that the same derivation extends to a full transformer block with attention and MLP layers; for simplicity, we focus here on the attention component alone, with the full derivation deferred to the Appendix A.5.

Consider a self-attention block applied to a signal with n_t timesteps, so the input has the form

$$X \in \mathbb{R}^{n_x \times n_t \times n_{in}},$$

where n_{in} is the input dimension and n_x is the batch size. The attention block first computes

$$Q = XW_Q^\top, \qquad K = XW_K^\top, \qquad Z = XW_Z^\top,$$

where we use Z for the value channel to avoid confusion with our weight-site notation V. The attention output is then

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{n_{attn}}}\right) Z, \qquad Y_{\mathrm{attn}} = AW_O^\top,$$

where $Q, K, Z, A \in \mathbb{R}^{n_x \times n_t \times n_{attn}}$.

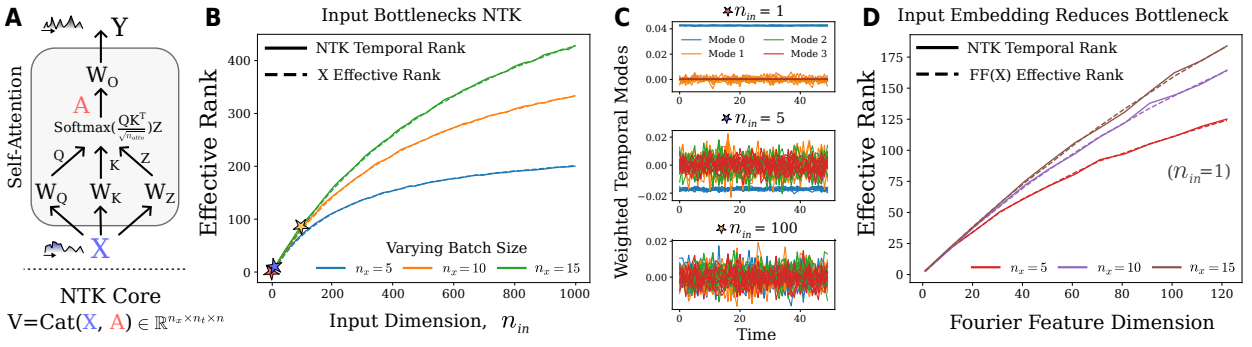

Figure 6: **Rank bottlenecking by input dimension in a self-attention model. A** Self-attention architecture with time-varying inputs $X \in \mathbb{R}^{n_x \times n_t \times n_{in}}$. As in the main text, the weight-site core of the NTK consists of the concatenated input activity and attention matrix, $V = \text{cat}(X, A)$. See Appendix A.2.3 for model and input details. **B** Because A lies in the same temporal span as X, the temporal rank of the NTK is bottlenecked by the batch size and input dimension. Increasing the input dimension increases the NTK temporal rank. **C** Three examples with distinct input dimension and fixed batch size $n_x = 10$, showing the variance-weighted dominant temporal modes of the NTK, $V_j \in \mathbb{R}^{n_x \times n_t}$. For one-dimensional inputs, there is only a single substantially non-zero mode, which is approximately constant over time. By contrast, for $n_{in} = 100$, there are many non-zero modes with a much flatter spectrum. Each mode is plotted over all 10 batch inputs, with color indicating mode identity. Here $X \sim \mathcal{N}(0, 1)$ is drawn randomly over time, batch, and input dimension, but the rank bottleneck applies for arbitrary task inputs. **D** Applying an input embedding (deterministic Fourier features in this case) maps a one-dimensional input signal into a higher-dimensional feature space. Increasing the number of features increases the NTK temporal rank (all of the curves correspond to $n_{in} = 1$, at the extreme left of the curves in panel B), reducing the bottleneck and yielding a more expressive range of GD updates.

Input Dimension Produces a Temporal Bottleneck The trainable weights are W_Q, W_K, W_Z, and W_O, so the corresponding weight sites are

$$V = \text{cat}(X, X, X, A).$$

Thus, by Theorem 1, the NTK_S core again has the form $VV^T \otimes I_n$, which can be viewed as a matrix of shape $n_x \cdot n_t$ by $3n_x + n_{attn}$. Naturally, this fits Theorem 1, with $k = n_x \cdot n_t$ constituting the *temporal dimension* of NTK_S, with partially reduced operator NTK_S^{temp} indicating dominant modes of this operator as signals over batches and input timesteps, and $n = 3n_x + n_{attn}$ the *spatial dimension* of the operator. In particular, since Q, K, and Z are all generated linearly from X, and A is formed by reweighting temporal content derived from these quantities, the dominant temporal modes available to NTK_S are strongly constrained by the temporal span already present in X.

This has the consequence that if the task input is low-dimensional, then the temporal rank of the NTK is necessarily low as well. Indeed, Figure 6B shows that as the input dimension increases, the temporal rank of the NTK increases with it. Figure 6C gives a more concrete view of this effect. For one-dimensional time-varying inputs, the NTK has essentially a single dominant temporal mode, which is nearly constant across time. For higher-dimensional inputs, many more temporal modes are available, and the spectrum becomes less concentrated. Additional sweeps in Appendix A.5.3 further clarify this distinction: while the temporal rank depends primarily on the input representation, the spatial rank and full operator rank grow strongly with attention width (Appendix Figure 9).

From the operator viewpoint, this means that gradient descent is not free to produce arbitrary temporal corrections in the attention model. Rather, it is constrained to update along the dominant temporal modes already present in the weight-site activity. If the input consists only of a narrow family of temporal signals, then the NTK is biased toward that same family. In particular, low-dimensional inputs induce a low-rank temporal bottleneck, so the resulting updates are confined to a restricted set of temporal patterns. This is a similar self-referential mechanism seen in the recurrent case, now arising from the representation bottleneck

of the attention block rather than from recurrent dynamics. At the same time, widening the attention block can still enlarge the spatial and overall rank of the operator, so the bottleneck here is specifically temporal rather than a claim that the full transformer NTK is globally low-rank.

Input Embeddings Relieve the Bottleneck In Figure 6D we show that the bottleneck can be relieved on a task of fixed input dimension by enriching the input representation before attention is applied. In our experiments, we use deterministic Fourier features to map a one-dimensional input into a higher-dimensional feature space. As the number of features increases, the temporal rank of the NTK increases as well, yielding a broader and more expressive range of GD updates. This provides a mechanistic explanation, within our Kronecker-core framework, for why richer input embeddings or positional features can substantially improve learnability in attention-based models. It is consistent with and effectively extends prior work, which used the infinite-width NTK for MLP models to show that Fourier features produce a better conditioned NTK for learning in this context, avoiding the inductive bias of GD toward learning low-frequency features of the target (Tancik et al., 2020). By contrast, adding multiple attention heads in this toy setting does not substantially change the same temporal bottleneck, consistent with the fact that the heads still derive their query, key, and value channels from the same underlying input representation (Appendix Figure 10).

Taken together, these results show that the Kronecker-core perspective extends beyond recurrence and sheds light on one reason richer input encodings can improve conditioning in transformer-like models. In a self-attention block, the temporal rank of NTK_S is bottlenecked by the structure of the input representation appearing in the weight sites, and gradient descent is correspondingly biased toward the dominant temporal modes of that representation. Enriching the input features lifts this temporal bottleneck and produces a more expressive temporal NTK, while widening the attention block primarily enriches the spatial and overall operator structure. Thus, even in the non-recurrent setting of transformer attention blocks, the core decomposition provides a concrete and interpretable explanation for low-rank, implicitly biased learning that can be overcome through positional embedding.

6 Discussion

We introduced a finite-width framework for analyzing gradient descent directly in the full model state space through the empirical NTK acting on the state tensor, NTK_S. From an implicit reformulation of the model dynamics. We show that

$$\mathrm{NTK}_S = \mathcal{P}\mathcal{K}\mathcal{P}^*,$$

The decomposition separates propagation through the model dynamics ($\mathcal{P}$) from immediate parameter-to-state interactions ($\mathcal{K}$). For a broad class of weight-based models, we then show that $\mathcal{K}$ has an exact Kronecker-core structure determined by forward-computable weight-site activity. This gives a tractable and interpretable handle on the state-space geometry of GD in the typical setting where NTK_S is otherwise too large to study directly. In particular, it makes clear why gradient descent is often biased toward low-rank, mode-selective solutions, a phenomenon we identify as self-referential bias. In this phenomenon, the model learns most easily along directions already present in its current hidden-input activity.

This perspective is useful both theoretically and practically. Theoretically, it gives a concrete language for describing how model structure, dynamics, and representation geometry constrain learning. Within the implicit formalism, it is natural to compute NTK_S for larger composed models by expanding the state or parameter spaces. One can also incorporate simplifying assumptions into the model by lifting the global constraint into a larger system of constraints that make the state or parameter structure more explicitly define. Then, computing $\mathcal{P}$ and $\mathcal{K}$ and projecting the NTK back to the original global state space yields an NTK reduced according to the assumption. This procedure is exactly used in the proof of Theorem 1. Finally, the framework makes the form of $\mathcal{P}$, as the inverse of a one-step generator, and $\mathcal{K}$, as a weight-site Gram matrix, explicit, allowing one to reason directly about how modifying dynamics or parameterization changes the bias of GD.

Practically, the approach provides a tool for diagnosis of bottlenecks. It can determine which tasks are accessible from a given initialization, when learning will be bottlenecked in space or time, and why some target components are learned quickly while others stall. More broadly, the framework provides a way to study

internal representation learning in finite-width networks without focusing the analysis on system outputs or needing to take infinite-width limits. The accompanying matrix-free toolkit, `kpflow`, makes these questions computationally accessible in real models.

A natural next step is to push this local theory further across training, to understand how these iteration-wise bottlenecks accumulate into global representation formation and persistent low-rank structure. This likely will require a stronger theory of the resolvent $\mathcal{P}$, perhaps by studying the inverse of the NTK, which avoids explicitly constructing $\mathcal{P}$. Under additional simplifying assumptions, the gradient flow and reachable domain of GD solutions may become tractable. For example, by truncating the Neumann series for $\mathcal{P}$ in Corollary 1 to a few terms in regimes of weak recurrence (Trousdale et al., 2012; Pernice et al., 2011).

Another important direction is to study how assumptions beyond rank constraints the structure of NTK_S. For example, if the network dynamics collapse to fixed points or pass through dynamical bifurcations, understanding the downstream effect on NTK_S could shed light on attractor formation hypotheses (Driscoll et al., 2024; Farrell et al., 2023; Qian and Pehlevan, 2025). These in turn are at the core of influential ideas of computation through dynamics and dynamical motifs (Vyas et al., 2020). It would be also useful to determine how structured perturbations or specialized synaptic plasticity rules, such as three-factor rules or other truncations of backpropagation, modify NTK_S (Lillicrap et al., 2016; Menick et al., 2020). Another interesting direction is to understand the impact of learning under $\mathcal{P}\mathcal{P}^*$ without the operator $\mathcal{K}$, matching a parameter-free, functional gradient update, or through modified forms of $\mathcal{K}$ that would follow from freezing a subset of the parameters or adding synaptic plasticity. In total, our framework opens a route toward studying how stability, latent structure, bifurcations, architectural constraints, parametrization, and the dimension and geometry of inputs geometry shape learnability in a wide range of practically used models.

7 References

References

Saman Alemohammad, Zhuang Wang, Remi Balestriero, and Richard Baraniuk. The recurrent neural tangent kernel. *Advances in Neural Information Processing Systems*, 33, 2020.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, 2019.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, 2019.

Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment, 2021. URL `https://arxiv.org/abs/2008.00938`.

Blake Bordelon and Cengiz Pehlevan. Dynamics of finite width kernel and prediction fluctuations in mean field neural networks. *arXiv preprint arXiv:2304.03408*, 2023.

Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning*, pages 1024–1034. PMLR, 2020.

Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. How feature learning can improve neural scaling laws. In *NeurIPS*, 2024.

Blake Bordelon, Jordan Cotler, Cengiz Pehlevan, and Jacob A. Zavatone-Veth. Dynamically learning to integrate in recurrent neural networks, 2025. URL `https://arxiv.org/abs/2503.18754`.

Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Out-of-distribution generalization in kernel regression, 2022. URL `https://arxiv.org/abs/2106.02261`.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. *arXiv preprint arXiv:1912.01198*, 2021.

Ricky T.Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

John P. Cunningham and Byron M. Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(11):1500–1509, November 2014. doi: 10.1038/nn.3776.

Laura N. Driscoll, Krishna V. Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, 27:1349–1363, 2024. doi: 10.1038/s41593-024-01668-6.

Weinan E. A proposal on machine learning via dynamical systems. *arXiv preprint arXiv:1711.00579*, 2017.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1 (3):211–218, 1936. doi: 10.1007/BF02288367.

Rainer Engelken, Fred Wolf, and L. F. Abbott. Lyapunov spectra of chaotic recurrent neural networks, 2020. URL `https://arxiv.org/abs/2006.02427`.

Zhou Fan and Zhichao Wang. Spectra of the conjugate kernel and neural tangent kernel for linear-width neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 7710–7721, 2020.

M. Farrell, S. Recanatesi, and E. Shea-Brown. From lazy to rich to exclusive task representations in neural networks and neural codes. *Current Opinion in Neurobiology*, 83:102780, 2023. doi: 10.1016/j.conb.2023.102780. Epub 2023 Sep 25.

Matthew Farrell, Stefano Recanatesi, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Gradient-based learning drives robust representations in recurrent neural networks by balancing compression and expansion. *Nature Machine Intelligence*, 4(6):564–573, 2022. doi: 10.1038/s42256-022-00498-0. URL `https://doi.org/10.1038/s42256-022-00498-0`.

Zhili Feng and J. Zico Kolter. On the neural tangent kernel of equilibrium models. *arXiv preprint arXiv:2310.14062*, 2023.

David J. Freedman and John A. Assad. Experience-dependent representation of visual categories in parietal cortex. *Nature*, 443(7107):85–88, September 2006. doi: 10.1038/nature05078.

Thomas George. NNGeometry: Easy and fast fisher information matrices and neural tangent kernels in PyTorch, February 2021. URL `https://doi.org/10.5281/zenodo.4532597`.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 573–582. PMLR, 2016.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018. URL `https://arxiv.org/abs/1802.09568`.

Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. In *International Conference on Learning Representations*, 2020.

James Hazelden. Fast neural tangent kernel alignment, norm and effective rank via trace estimation, 2025. URL `https://arxiv.org/abs/2511.10796`.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

Darko Janeković and Dario Bojanjac. Randomized algorithms for singular value decomposition: Implementation and application perspective. In *2021 International Symposium ELMAR*, pages 165–168, 2021. doi: 10.1109/ELMAR52657.2021.9550979.

Yann LeCun. A theoretical framework for back-propagation. In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 21–28. Morgan Kaufmann, San Mateo, CA, 1988.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124002, December 2020. doi: 10.1088/1742-5468/abc62b. URL `https://doi.org/10.1088/1742-5468/abc62b`.

Richard B. Lehoucq, Danny C. Sorensen, and Chao Yang. *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, PA, 1998.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:13276, November 2016. doi: 10.1038/ncomms13276.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 2408–2417, 2015.

Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623.e29, 2018. doi: 10.1016/j.neuron.2018.07.003.

Dana Mastrovito, Yuhan Helena Liu, Lukasz Kusmierz, Eric Shea-Brown, Christof Koch, and Stefan Mihalas. Transition to chaos separates learning regimes and relates to measure of consciousness in recurrent neural networks. *bioRxiv*, page 2024.05.15.594236, may 2024. doi: 10.1101/2024.05.15.594236. Preprint, Version 1.

Song Mei, Tomasz Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layer neural networks: dimension-free bounds and kernel limit. *Advances in Neural Information Processing Systems*, 31, 2018.

Jacob Menick, Erich Elsen, Utku Evci, Simon Osindero, Karen Simonyan, and Alex Graves. A practical sparse approximation for real time recurrent learning. *arXiv preprint arXiv:2006.07232*, June 2020.

Raphael A. Meyer, Cameron Musco, Christopher Musco, and David P. Woodruff. Hutch++: Optimal stochastic trace estimation, 2021. URL `https://arxiv.org/abs/2010.09649`.

Mohamad Amin Mohamadi, Wonho Bae, and Danica J. Sutherland. A fast, well-founded approximation to the empirical neural tangent kernel. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 25011–25035, 2023.

Guillermo B. Morales, Serena Di Santo, and Miguel A. Muñoz. Unveiling the intrinsic dynamics of biological and artificial neural networks: from criticality to optimal representations. *arXiv preprint arXiv:2307.10669*, July 2023.

Vladislav Myrov, Alina Suleimanova, Samanta Knapič, Paula Partanen, M. Vesterinen, W. Liu, S. Palva, and J. M. Palva. Hierarchical whole-brain modeling of critical synchronization dynamics in the human brain. *Proceedings of the National Academy of Sciences*, 123(12):e2505768123, 2026. doi: 10.1073/pnas.2505768123. URL `https://doi.org/10.1073/pnas.2505768123`.

Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition edition, 2010.

Roman Novak, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Fast finite width neural tangent kernel, 2022. URL https://arxiv.org/abs/2206.08720.

Srdjan Ostojic and Stefano Fusi. Computational role of structure in neural activity and connectivity. *Trends in Cognitive Sciences*, 28(7):677–690, 2024. doi: 10.1016/j.tics.2024.03.003.

Barak A. Pearlmutter. Gradient calculations for dynamic systems. In *Automatic Differentiation of Algorithms*, pages 198–222. Springer, 1995.

Arthur Pellegrino, N. Alex Cayco Gajic, and Angus Chadwick. Low tensor rank learning of neural dynamics. In *Advances in Neural Information Processing Systems*, volume 36, pages 11674–11702, 2023.

Volker Pernice, Benjamin Staude, Stefano Cardanobile, and Stefan Rotter. How structure determines correlations in neuronal networks. *PLoS Computational Biology*, 7(5):e1002059, May 2011. doi: 10.1371/journal.pcbi.1002059.

L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. *The Mathematical Theory of Optimal Processes*. Pergamon Press, 1962. Translated from the Russian original.

William Qian and Cengiz Pehlevan. Discovering alternative solutions beyond the simplicity bias in recurrent neural networks. *arXiv preprint arXiv:2509.21504*, 2025. doi: 10.48550/arXiv.2509.21504.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A. Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Kanaka Rajan and L. F. Abbott. Eigenvalue spectra of random matrices for neural networks. *Physical Review Letters*, 97(18):188104, 2006. doi: 10.1103/PhysRevLett.97.188104.

Kanaka Rajan, L. F. Abbott, and Haim Sompolinsky. Stimulus-dependent suppression of chaos in recurrent neural networks. *Physical Review E*, 82(1):011903, 2010.

Stefano Recanatesi, Matthew Farrell, Guillaume Lajoie, Sophie Deneve, Mattia Rigotti, and Eric Shea-Brown. Predictive learning as a network mechanism for extracting low-dimensional latent space representations. *Nature Communications*, 12(1):1417, 2021. doi: 10.1038/s41467-021-21696-1. URL https://doi.org/10.1038/s41467-021-21696-1.

Stefano Recanatesi, Serena Bradde, Vijay Balasubramanian, Nicholas A. Steinmetz, and Eric Shea-Brown. A scale-dependent measure of system dimensionality. *Patterns*, 3(8):100555, August 2022. ISSN 2666-3899. doi: 10.1016/j.patter.2022.100555.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Friedrich Schuessler, Francesca Mastrogiuseppe, Alexis Dubreuil, Srdjan Ostojic, and Omri Barak. The interplay between randomness and structure during learning in rnns. In *Advances in Neural Information Processing Systems*, volume 33, pages 13352–13362. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/9ac1382fd8fc4b631594aa135d16ad75-Paper.pdf.

Haozhe Shan and Blake Bordelon. A theory of neural tangent kernel alignment and its influence on training, 2022. URL https://arxiv.org/abs/2105.14301.

Haim Sompolinsky, Andrea Crisanti, and Hermann J. Sommers. Chaos in random neural networks. *Physical Review Letters*, 61(3):259–262, 1988. doi: 10.1103/PhysRevLett.61.259.

David Sussillo and L. F. Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009. doi: 10.1016/j.neuron.2009.07.018.

David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013. doi: 10.1162/NECO_a_00409.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020. URL https://arxiv.org/abs/2006.10739.

James Trousdale, Yu Hu, Eric Shea-Brown, and Krešimir Josić. Impact of network structure and cellular response on spike time correlations. *PLoS Computational Biology*, 8(3):e1002408, March 2012. doi: 10.1371/journal.pcbi.1002408.

Elia Turner and Omri Barak. The simplicity bias in multi-task rnns: Shared attractors, reuse of dynamics, and geometric representation. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 25495–25507. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/50d6dbc809b0dc96f7f1090810537acc-Paper-Conference.pdf.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jimmy Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Millman, Nikolay Mayorov, Andrew Nelson, Eric Jones, Robert Kern, Eric Larson, Craig J. Carey, İsmail Polat, Yu Feng, Eric W. Moore, S. Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, Alexandre H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

Saurabh Vyas, Matthew D. Golub, David Sussillo, and Krishna V. Shenoy. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43:249–275, July 2020. doi: 10.1146/annurev-neuro-092619-094115.

Paul J. Werbos. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78 (10):1550–1560, 1990. doi: 10.1109/5.58337.

Jinlin Xiang, Minho Choi, Yubo Zhang, Zhihao Zhou, Arka Majumdar, and Eli Shlizerman. Neural tangent knowledge distillation for optical convolutional networks, 2025. URL https://arxiv.org/abs/2508.08421.

Greg Yang. Tensor programs ii: Neural tangent kernel for any architecture. *arXiv preprint arXiv:2006.14548*, 2020.

Greg Yang and Edward J. Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11727–11737. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/yang21c.html.

Guangyu Robert Yang, Maitham R. Joglekar, Henry F. Song, William T. Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature Neuroscience*, 22 (2):297–306, 2019. doi: 10.1038/s41593-018-0310-2.

A Appendix

A.1 The `kpflow` Package for Analysis of the Empirical Global NTK

Here we detail the backend implementation of the operators used throughout the paper. Some details, including timing and implementation aspects of the randomized trace estimation routines described below, are discussed more fully in a prior preprint (Hazelden, 2025).

A.1.1 High-Level Structure

Matrix-free interface. Every operator in the paper is implemented as a derived class of an `Operator` base class. Importantly, all operators are implemented in a matrix-free way. Concretely, an `Operator` exposes only two methods implemented by the derived class: `matvec` and `rmatvec`. For an operator $\mathcal{G} : S_1 \to S_2$, these correspond to application of $\mathcal{G}$ and $\mathcal{G}^*$, respectively, to candidate vectors in S_1 or S_2. Thus all analysis of the operators is carried out by probing them with input vectors $v \in S$, rather than explicitly forming their full matrices.

Fortunately, there is a substantial literature on matrix-free numerical linear algebra for exactly this setting. Many such methods are randomized, estimating operator-level quantities by sketching or probing the action of the matrix on random vectors. Using these ideas, the base `Operator` class supports a range of analysis routines, including estimation of the top singular modes, alignment, effective rank, and operator norms. In addition, we expose an interface that maps closely to the theory, allowing one to compose operators, compute tensor products, compare against exact matrix representations when tractable, and form reduced partial-trace views, as described below.

Estimating the trace through randomized algorithms. One of the basic quantities exposed for every operator is its trace, i.e. the sum of its eigenvalues (equivalently, diagonal entries when the operator is represented as a matrix). To estimate this efficiently, we use Hutch++, which can provide accurate trace estimates with relatively few probes, especially when the operator is approximately low-rank (Hazelden, 2025; Meyer et al., 2021). If the operator is not low-rank, performance degrades, but, as discussed in the main paper, the NTK is often strongly low-rank in the settings studied here, so Hutch++ is well suited to this use case.

Trace-derived quantities. The trace is also used as a basic building block for other analysis metrics. In particular, it yields approximations to an operator's effective rank, Frobenius norm, and cosine similarity with another operator on the same domain.

Randomized SVD. One of the most informative descriptors of an operator is its leading singular structure. Matrix-free methods for estimating the top singular components are well developed (Janeković and Bojanjac, 2021). In our implementation, we rely on `scipy`'s `LinearOperator` interface together with sparse eigensolvers such as `eigsh` for symmetric operators (Virtanen et al., 2020; Lehoucq et al., 1998).

A.1.2 Tensor Operator Interface

Composing operators. Two operators in `kpflow` can be combined in multiple ways. Given operators $\mathcal{G}_1 : S_1 \to S_2$ and $\mathcal{G}_2 : S_2 \to S_3$, their composition $\mathcal{G}_2 \circ \mathcal{G}_1 : S_1 \to S_3$ corresponds to the product of their explicit matrices. In our package, the syntax to compose two operators `op1` and `op2` is `op1 @ op2`. Another way to combine operators is by the tensor product $\mathcal{G}_1 \otimes \mathcal{G}_2$, for arbitrary $\mathcal{G}_1 : S_1 \to S_2$ and $\mathcal{G}_2 : S_3 \to S_4$. The resulting operator $\mathcal{G}_1 \otimes \mathcal{G}_2 : S_1 \otimes S_3 \to S_2 \otimes S_4$ acts on tensor-product inputs. In code, this is exposed by the syntax `op1.tprod(op2)`.

Example syntax. Basic operators such as the identity and wrappers for dense numpy matrices are also implemented. For example, if `kop` denotes the operator $\mathcal{K}$ from Proposition 1, and V is the $k \times m$ weight-site matrix from Theorem 1, then the theorem can be checked numerically by verifying that the relative operator error is small:

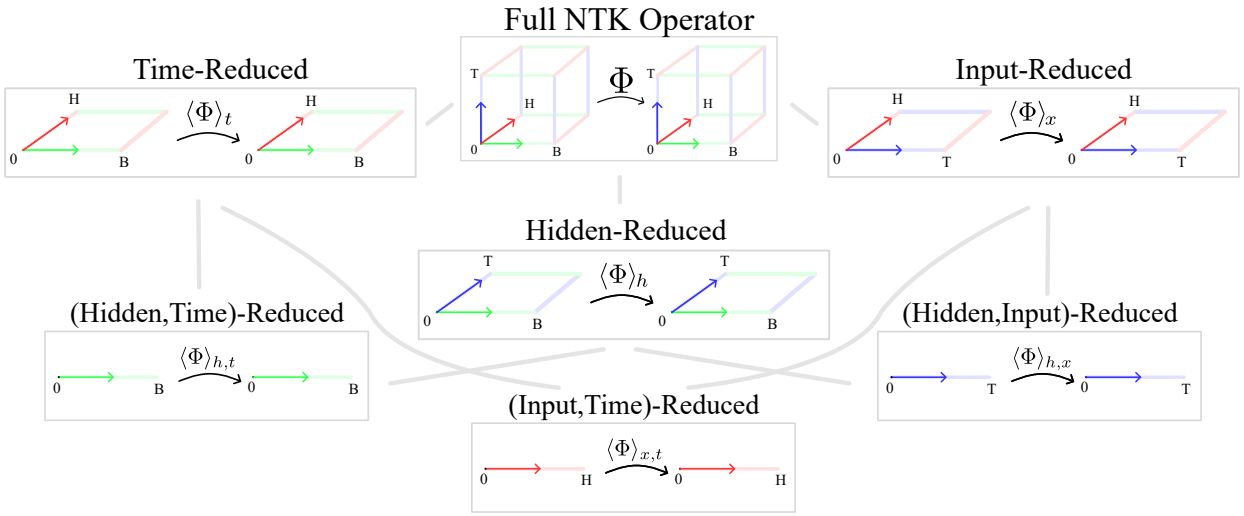

Figure 7: **Examples of partial reductions of an operator acting on a 3-tensor domain, $\mathbb{R}^{B\times T\times H}$.** The operator Φ acts on a domain $\mathbb{R}^{B\times T\times H}$, representing batch, time, and hidden-unit axes. Each reduction averages over one or more axes. The reduced operators at the bottom are simple $B\times B$, $T\times T$, and $H\times H$ matrices, respectively. These capture how the operator varies across batch inputs, hidden units, or timesteps after averaging over the complementary axes.

```
err = kop.compare(MatrixWrapper(V @ V.T).tprod_like(IdentityOperator(n), kop))      (16)
```

Here `compare` computes a relative error between two operators. The wrapper `MatrixWrapper` converts the dense matrix $VV^\top$ into an operator. The function `tprod_like` forms the tensor product with the supplied operator and reshapes the result to match the domain and codomain structure of the reference operator, here `kop`. Finally, `IdentityOperator(n)` constructs the $n\times n$ identity operator. In total, this uses randomized Frobenius norm estimation under the hood to verify numerically that

$$\mathcal{K}\approx VV^\top\otimes I_n.$$

A.1.3 Partial Trace Views

The partial trace. The operators discussed in the paper act naturally on tensor domains. It is common to vectorize such domains and ignore this structure, but keeping the tensor structure explicit yields more informative reduced views. For example, the recurrent NTK operator from Corollary 1 acts on 3-tensors in $\mathbb{R}^{n_x\times n_t\times n}$. Equivalently, the operator itself can be viewed as a 6-tensor. A useful tool for analyzing such an object is the *partial trace*, which traces over one or more tensor axes.

As in the main text, we define reduced operators such as $\texttt{ntk}_{\text{temp}}$, which averages over the hidden dimension n and yields an operator acting on the $n_x\times n_t$ axes, and $\texttt{ntk}_{\text{space}}$, which averages over the batch and time axes and yields an $n\times n$ matrix. The effective ranks of these reduced operators are used in Figure 5 of the main text.

In code, the partial trace is implemented, for example, by

```
ntk_temp = ntk.partial_avg(2),
```

which averages over the final axis. Under the hood, this is done in a matrix-free way: (1) inputs to the reduced operator are expanded across the traced axis, (2) passed through the original operator, and (3) the

outputs are averaged over that same axis. Concretely, for $x \in \mathbb{R}^{n_x \cdot n_t}$,

$$\mathrm{NTK}_S^{\mathrm{temp}}(x) = \frac{1}{n} \sum_{j=1}^{n} \mathrm{NTK}_S(x e_j^\top),$$

where the matrix $x e_j^\top$ corresponds to expanding x into the missing hidden dimension.

These reduced operators can then be combined with SVD routines to compute reduced singular modes, for example dominant time-by-trial modes after averaging over hidden units, as in Figure 3 in the main text. For an operator acting on $\mathbb{R}^{B \times T \times H}$, there are $2^3 - 1 = 7$ nontrivial partial reductions, giving a family of complementary views of how the operator acts across different axes.

Forming explicit reduced matrices. Finally, once an operator has been reduced enough, it can become tractable to form explicitly. For example, $\texttt{ntk}_{\mathrm{space}}$ is an $n \times n$ matrix, where n is the hidden dimension, and so can often be constructed exactly. To do this, we provide a method `op.full_matrix()` that forms the corresponding dense matrix. For instance, the batch-by-batch reduced NTK can be computed via

```
ntk_batches = ntk.partial_avg((1,2)).full_matrix()
```

which averages over time and hidden dimensions before explicitly forming the reduced operator. Thus, although treating the NTK as a tensor operator is conceptually more involved, it naturally yields multiple informative reductions tailored to different questions.

A.2 Experimental Appendix

Here we give additional details for the experiments in the main text.

A.2.1 GRU Memory Pro

Constructing the 5-fixed-point initialization (Network 2). We use $n = 256$ neurons in these experiments. The goal is to construct a recurrent initialization with multiple nontrivial fixed points at initialization, and then use the same idea to initialize the GRU recurrent weights.

Specifically, consider an RNN in the response period of the Memory-Pro task, where the task input is zero. We want to construct an RNN with nontrivial fixed points, rather than having all task-driven trajectories collapse to the trivial fixed point, as happens under a default Xavier initialization. Consider the update

$$h_{t+1} = W\sigma(h_t).$$

Then

$$h_{t+1} - h_t = W\sigma(h_t) - h_t.$$

Setting this to zero at a desired fixed point $\bar{h}$ gives

$$W\sigma(\bar{h}) - \bar{h} = 0.$$

This is achieved by any W of the form

$$W = \frac{\bar{h}_1 \sigma(\bar{h}_1)^\top}{\|\sigma(\bar{h}_1)\|^2} + W_1,$$

where W_1 is any matrix that maps $\sigma(\bar{h}_1)$ to zero.

More generally, if $\sigma(\bar{h}_1), \ldots, \sigma(\bar{h}_m)$ are orthogonal, then we can construct a matrix W with all of these points fixed:

$$W = \sum_{i=1}^{m} \frac{\bar{h}_i \sigma(\bar{h}_i)^\top}{\|\sigma(\bar{h}_i)\|^2} + W_{1:m}, \tag{17}$$

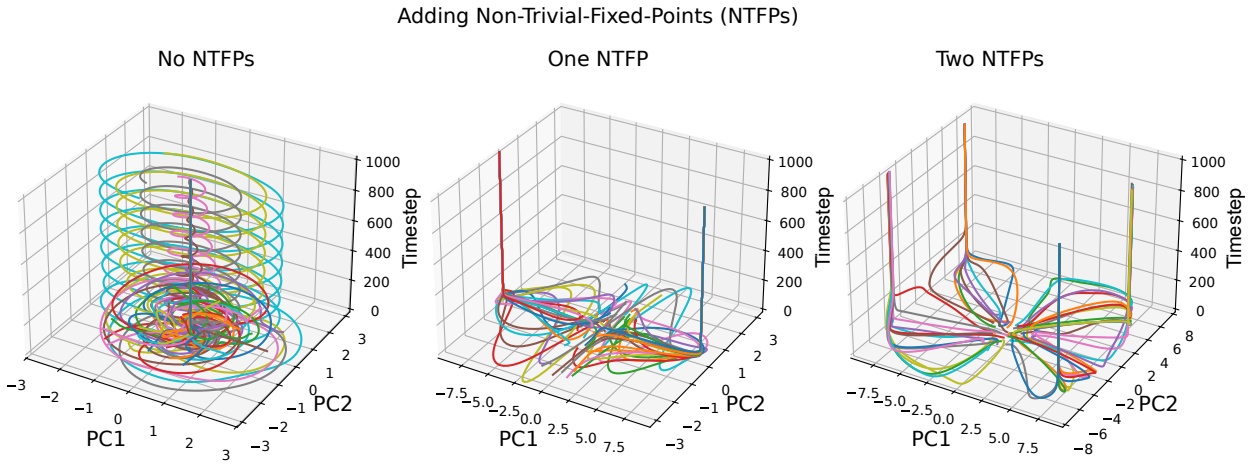

Figure 8: **Dynamics of an RNN with added non-trivial fixed points (NTFPs).** Random weights are sampled with Xavier initialization, scaled by gain g, with $n = 256$ and g varied between 1 and 2. Non-trivial fixed points are then added to the model as described in the text. Plotted trajectories correspond to distinct random initial conditions and gain values. The random seed is fixed between trials and the dynamics are projected into a shared PCA space. Time is shown on the vertical axis. In the absence of added NTFPs, the RNN transitions between collapse and chaotic behavior. For small gain, trajectories collapse to the trivial fixed point at zero; for larger gain, the dynamics become unstable. With one added NTFP, the system becomes effectively bistable under all gains shown. With two added NTFPs, the dynamics collapse to four distinct fixed points. The number of effective fixed points doubles because tanh is odd.

where $W_{1:m}$ annihilates each $\sigma(\bar{h}_i)$. Indeed,

$$W\sigma(\bar{h}_j) = \sum_{i=1}^{m} \frac{\bar{h}_i \, \delta_{ij} \, \|\sigma(\bar{h}_i)\|^2}{\|\sigma(\bar{h}_i)\|^2} + 0 = \bar{h}_j. \tag{18}$$

A natural choice is to take the $\bar{h}_i$ to be scaled orthogonal coordinate directions, since $\sigma(ke_j) = \sigma(k)e_j$ for coordinate vectors when σ acts componentwise. In that case the orthogonality of the transformed vectors is immediate.

Task details. We use fixed-duration stimulus, memory, and response periods so that the hidden-state and output dynamics are easier to interpret. Each trial has $n_t = 90$ timesteps, with 30 timesteps per task phase. We use batch size 500, with 5,000 total task inputs and targets. During training, Gaussian noise with variance 3.2 is added to the inputs to encourage more robust attractor structure; this noise is turned off during evaluation and plotting (Driscoll et al., 2024; Qian and Pehlevan, 2025). The learning rate is 10^{-3}.

A.2.2 Vanilla RNN Space-Time NTK Rank

Task setup. For the spatial and temporal regime experiments in Section 5.3, we use a nonlinear student–teacher RNN setting. The teacher and student share the same architecture and are both driven by i.i.d. standard normal input sequences. The teacher is initialized with a fixed recurrent gain g^*, which is set to $g^* = 1$ in all experiments. The student uses the same architecture, but one subset of weights is re-initialized.

In the recurrent-weight task, the student's input and output weights are set equal to the teacher's, while the recurrent weights W are re-sampled with Xavier initialization and gain g. The goal is then to recover the teacher output by training only W. In the input-weight task, the student's recurrent and output weights are instead matched to the teacher, while the input weights W_{in} are re-sampled with Xavier initialization and gain g, and only these weights are trained. In both cases the task is realizable by returning the perturbed

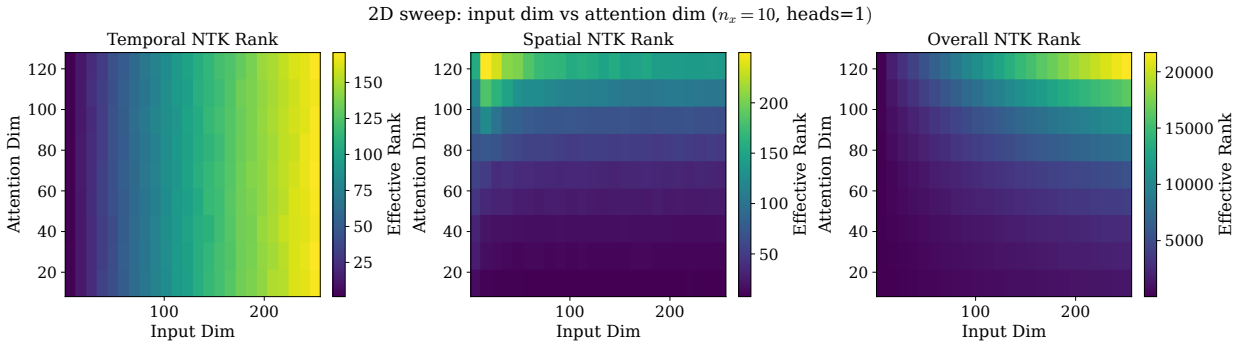

Figure 9: **Temporal, spatial, and overall NTK effective rank for a two-dimensional sweep over input dimension and attention dimension.** The temporal rank grows primarily with input dimension, while the spatial and overall rank increase strongly with attention width.

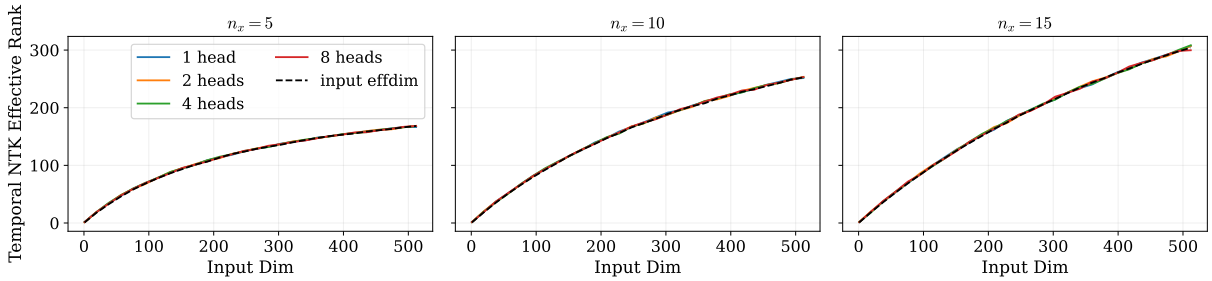

Figure 10: **Temporal NTK rank versus input dimension for a single-block self-attention toy model with varying head count.** Increasing the number of attention heads has only a modest effect on temporal rank, so the same basic temporal bottleneck from the input representation remains.

weights to the teacher values, so slow or incomplete learning reflects bias in the induced NTK rather than task misspecification.

Unless otherwise stated, we use $n_{\text{in}} = 16$ input channels, $n = 64$ hidden units, $n_{\text{out}} = 1$ output channel, and $n_t = 40$ timesteps, with an MSE loss applied at every timestep. Models are trained with vanilla SGD using batch size 128, learning rate 0.01, and no momentum or gradient clipping.

A.2.3 Transformer Global-State NTK

Transformer setup. For the transformer experiment, we use a single self-attention block followed by an MLP, evaluated at initialization on i.i.d. Gaussian input sequences $X \in \mathbb{R}^{n_x \times n_t \times n_{\text{in}}}$. We vary $n_x \in \{5, 10, 15\}$, use attention width $n_{\text{attn}} = 64$, $n_{\text{out}} = 1$, and $n_t = 50$, and take the MLP to have 3 layers with 128 neurons per layer. We then vary the input dimension n_{in} and measure the temporal rank of the resulting global-state NTK as a function of input dimension and batch size. The relevant weight-site matrix is computed from the transformer and MLP weight-sites, consisting of the projected input, attention output, and hidden activations of the MLP (excluding the final readout layer), as derived in Appendix A.5.

To test whether this bottleneck can be relieved, we also apply a deterministic one-dimensional Fourier-feature embedding (Tancik et al., 2020) before the input projection and sweep the number of Fourier frequencies. As the feature dimension increases, both the effective rank of the embedded input and the temporal rank of NTK_S increase.

Additional appendix sweeps support the same interpretation. Varying the number of attention heads changes the temporal rank only modestly across input dimensions, so multi-head attention does not remove the

same basic temporal bottleneck in this toy setting (Figure 10). By contrast, a two-dimensional sweep over input dimension and attention width shows a clearer separation: the temporal rank grows primarily with input dimension, whereas the spatial and overall ranks grow strongly with attention width (Figure 9). Thus, widening attention mainly enriches the spatial and overall operator structure, while the temporal expressivity of NTK_S remains bottlenecked by the input representation.

A.3 Proof of Theorem 1

A.3.1 Weight-Based Model Definition

A weight-based model is one in which the parameters θ can be represented, after lifting if necessary, by a global matrix $W \in \mathbb{R}^{n\times m}$ that is applied to internal *weight-site* variables collected in a matrix $V \in \mathbb{R}^{k\times m}$. Letting

$$Q := VW^\top \in \mathbb{R}^{k\times n},$$

we assume the model can be written in the form

$$\mathcal{F}(h, x, \theta) = \mathcal{G}_h(h, V, Q, x),$$

where V, Q, W are uniquely determined by a lifted system of constraints in the augmented state space (h, V, Q). Concretely, the lifted system has the form

$$0 = \mathcal{F}_{\mathrm{aug}}((h, V, Q), x, W) = \begin{pmatrix} \mathcal{G}_h(h, V, Q, x) \\ \mathcal{G}_V(h, V, Q, x) \\ Q - VW^\top \end{pmatrix}, \tag{19}$$

where $\mathcal{G}_V(h, V, Q, x) = 0$ determines the weight-site variables V, and the final constraint enforces $Q = VW^\top$. We assume $\mathcal{G}_h$ maps into S, and $\mathcal{G}_V$ maps into $\mathbb{R}^{k\times m}$.

This definition guarantees that the parameter dependence enters through matrix–vector interactions with the weight sites. Allowing V to be defined implicitly rather than explicitly in terms of h and x accommodates nested architectures, where intermediate weight-site variables themselves depend on earlier weighted computations.

Infinite weight-site case. More generally, one may allow infinitely many weight sites ($k = \infty$), but we avoid that notation here for simplicity. The proof is unchanged in essence: the object $VV^\top$ is then interpreted as a second-moment operator constructed by probing, rather than as an explicitly formed matrix. Its rank is still bounded by the feature dimension m, so a finite number of probes suffices to identify its range.

Motivating nested example. Consider the nested map $y = W_2\sigma(W_1x)$. Define

$$v_1 := x, \qquad q_1 := W_1v_1, \qquad v_2 := \sigma(q_1), \qquad q_2 := W_2v_2,$$

so that $y = q_2$. Stacking the weights into a block-diagonal matrix $W = \mathrm{blockdiag}(W_1, W_2)$, the weight sites can be written as

$$V = \begin{pmatrix} x & 0 \\ 0 & \sigma(q_1) \end{pmatrix}, \qquad Q = VW^\top.$$

This can be formulated as a lifted system of constraints in (y, V, Q). The point is that even nested architectures can be rewritten so that all parameter dependence appears through products of a shared weight matrix with weight-site variables.

A.3.2 Proof

Proof. Lift the model into the augmented state variables $W \in \mathbb{R}^{n\times m}$, $V \in \mathbb{R}^{k\times m}$, and $Q \in \mathbb{R}^{k\times n}$ using the constraint system in Equation equation 19:

$$0 = \mathcal{F}_{\mathrm{aug}}((h, V, Q), x, W) = \begin{pmatrix} \mathcal{G}_h(h, V, Q, x) \\ \mathcal{G}_V(h, V, Q, x) \\ Q - VW^\top \end{pmatrix}.$$

By construction, the original constraint satisfies

$$\mathcal{F}(h, x, \theta) = \mathcal{G}_h(h, V, Q, x),$$

with V and Q uniquely determined by the lifted system at the current state.

Define the augmented state space

$$S_{\text{aug}} := S \oplus \mathbb{R}^{k\times m} \oplus \mathbb{R}^{k\times n}, \qquad h_{\text{aug}} := (h, V, Q) \in S_{\text{aug}}.$$

Applying Proposition 1 to the lifted system yields

$$\mathcal{P}_{\text{aug}} := (D_{h_{\text{aug}}}\mathcal{F}_{\text{aug}})^{-1}, \qquad \mathcal{K}_{\text{aug}} := (D_W\mathcal{F}_{\text{aug}})(D_W\mathcal{F}_{\text{aug}})^*.$$

The corresponding first-order correction in the lifted state is

$$\delta h_{\text{aug}} = \begin{pmatrix} \delta h \\ \delta V \\ \delta Q \end{pmatrix} = \text{NTK}_{S_{\text{aug}}} \begin{pmatrix} \text{err}(h) \\ 0 \\ 0 \end{pmatrix}.$$

If we view $\mathcal{P}_{\text{aug}}$ as a 3×3 block operator, then the correction in the original state h is obtained from the first row:

$$\delta h = \mathcal{P}_{\text{aug},1}\, \mathcal{K}_{\text{aug}}\, \mathcal{P}^*_{\text{aug},1}(\text{err}(h)).$$

Since $\text{err}(h)$ is arbitrary, this shows that the original NTK can be written as

$$\text{NTK}_S = \mathcal{P}_{\text{aug},1}\, \mathcal{K}_{\text{aug}}\, \mathcal{P}^*_{\text{aug},1}.$$

Next, we compute $\mathcal{K}_{\text{aug}}$. Since only the final block $Q - VW^\top$ depends on W,

$$D_W\mathcal{F}_{\text{aug}} = \begin{pmatrix} 0 \\ 0 \\ -(V \otimes I_n) \end{pmatrix}.$$

Therefore

$$\mathcal{K}_{\text{aug}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & VV^\top \otimes I_n \end{pmatrix}.$$

Hence only the $(1, 3)$ block of $\mathcal{P}_{\text{aug}}$ contributes to the induced NTK on S, giving

$$\text{NTK}_S = \mathcal{P}_{\text{aug},1,3}\, (VV^\top \otimes I_n)\, \mathcal{P}^*_{\text{aug},1,3}.$$

This is the desired factorization. The matrix $VV^\top$ is the Gram matrix of the weight-site state, so it acts as an unnormalized projector onto the dominant modes already present in the weight sites. This is the Kronecker-core bottleneck described in the theorem.

Existence of $\mathcal{P}_{\text{aug},1,3}$. For completeness, we justify that the block $\mathcal{P}_{\text{aug},1,3}$ is well defined. Explicitly,

$$D_{h_{\text{aug}}}\mathcal{F}_{\text{aug}} = \begin{pmatrix} D_h\mathcal{G}_h & D_V\mathcal{G}_h & D_Q\mathcal{G}_h \\ D_h\mathcal{G}_V & D_V\mathcal{G}_V & D_Q\mathcal{G}_V \\ 0 & -(W \otimes I_k) & I_{kn} \end{pmatrix}.$$

The invertibility of this block operator reduces to invertibility of the corresponding Schur complement

$$C := D_h\mathcal{G}_h - \big[D_V\mathcal{G}_h + D_Q\mathcal{G}_h(W \otimes I_k)\big]\big[D_V\mathcal{G}_V + D_Q\mathcal{G}_V(W \otimes I_k)\big]^{-1} D_h\mathcal{G}_V.$$

But by construction $\mathcal{F}(h, x, \theta) = \mathcal{G}_h(h, V, Q, x)$, so the chain rule gives

$$D_h\mathcal{F} = D_h\mathcal{G}_h + D_V\mathcal{G}_h\, D_hV + D_Q\mathcal{G}_h\, D_hQ.$$

Since $Q = VW^\top$ and $\mathcal{G}_V(h, V, Q, x) = 0$ determines V, the implicit function theorem shows that this Jacobian is exactly the Schur complement C. Therefore

$$D_h\mathcal{F} = C.$$

By assumption, the original operator $\mathcal{P} = (D_h\mathcal{F})^{-1}$ exists, so C is invertible. Hence $\mathcal{P}_{\text{aug}}$ exists, and in particular $\mathcal{P}_{\text{aug},1,3}$ is well defined.

Writing the corresponding block inverse explicitly gives

$$\mathcal{P}_{\text{aug},1,3} = \mathcal{P}\left(D_h\mathcal{F} + \left[D_V\mathcal{G}_h + D_Q\mathcal{G}_h(W \otimes I_k)\right]\left[D_V\mathcal{G}_V + D_Q\mathcal{G}_V(W \otimes I_k)\right]^{-1} D_Q\mathcal{G}_V\right). \tag{20}$$

The precise form is not especially important in applications; what matters is that it exists and induces the factorization

$$\text{NTK}_S = \mathcal{P}_{\text{core}}(VV^\top \otimes I_n)\mathcal{P}_{\text{core}}^*, \qquad \mathcal{P}_{\text{core}} := \mathcal{P}_{\text{aug},1,3}.$$

In practice, one usually identifies the weight sites directly from the architecture, rather than computing Equation equation 20 explicitly. □

A.4 Proof of Proposition 2

Proof. For a broader discussion, including Schmidt decompositions of operators over tensor-product spaces, see (Nielsen and Chuang, 2010; Eckart and Young, 1936). Here we only need the fact that the NTK is positive semidefinite, since it is a Gram operator.

We first show that the reduced views are also positive semidefinite. Let $(e_j)_{j=1}^n$ be an orthonormal basis of $\mathbb{R}^n$, and let $u \in \mathbb{R}^k$. Then

$$\langle u, \text{NTK}_S^{\text{temp}}(u)\rangle = \frac{1}{n}\sum_{j=1}^{n}\langle ue_j^\top, \text{NTK}_S(ue_j^\top)\rangle_F.$$

Each term in the sum is nonnegative because NTK_S is PSD, hence $\text{NTK}_S^{\text{temp}}$ is PSD. The same argument shows that $\text{NTK}_S^{\text{space}}$ is PSD.

Now let $E \in \mathbb{R}^{k\times n}$ be any error signal, and let $\delta h = \text{NTK}_S(E)$. We will show

$$\text{Im}(\text{NTK}_S(E)) \subseteq \text{Im}(\text{NTK}_S^{\text{temp}}), \tag{F1}$$

and

$$\text{Im}(\text{NTK}_S(E)^\top) \subseteq \text{Im}(\text{NTK}_S^{\text{space}}). \tag{F2}$$

These two inclusions imply

$$\text{rank}(\delta h) \le \min\big(\text{rank}(\text{NTK}_S^{\text{temp}}), \text{rank}(\text{NTK}_S^{\text{space}})\big),$$

since row and column rank are equal.

We now prove F1. Let $u \in \ker(\text{NTK}_S^{\text{temp}})$. Since $\text{NTK}_S^{\text{temp}}$ is PSD,

$$0 = \langle u, \text{NTK}_S^{\text{temp}}(u)\rangle = \frac{1}{n}\sum_{j=1}^{n}\langle ue_j^\top, \text{NTK}_S(ue_j^\top)\rangle_F.$$

Each term is nonnegative, so each term must in fact be zero:

$$\langle ue_j^\top, \text{NTK}_S(ue_j^\top)\rangle_F = 0, \qquad j = 1, \dots, n.$$

Because NTK_S is PSD, this implies

$$\text{NTK}_S(ue_j^\top) = 0, \qquad j = 1, \dots, n.$$

Using symmetry of NTK_S, for any j,

$$\langle u, \mathrm{NTK}_S(E)e_j\rangle = \langle ue_j^\top, \mathrm{NTK}_S(E)\rangle_F = \langle \mathrm{NTK}_S(ue_j^\top), E\rangle_F = 0.$$

Therefore $u^\top \mathrm{NTK}_S(E) = 0$. Since this holds for all $u \in \ker(\mathrm{NTK}_S^{\mathrm{temp}})$, we conclude

$$\mathrm{Im}(\mathrm{NTK}_S(E)) \subseteq \ker(\mathrm{NTK}_S^{\mathrm{temp}})^\perp.$$

Because $\mathrm{NTK}_S^{\mathrm{temp}}$ is symmetric,

$$\ker(\mathrm{NTK}_S^{\mathrm{temp}})^\perp = \mathrm{Im}(\mathrm{NTK}_S^{\mathrm{temp}}),$$

which proves F1.

The proof of F2 is identical after exchanging the temporal and spatial roles. If $v \in \ker(\mathrm{NTK}_S^{\mathrm{space}})$, then the same argument shows

$$\mathrm{NTK}_S(E)v = 0.$$

Hence

$$\mathrm{Im}(\mathrm{NTK}_S(E)^\top) \subseteq \ker(\mathrm{NTK}_S^{\mathrm{space}})^\perp = \mathrm{Im}(\mathrm{NTK}_S^{\mathrm{space}}).$$

This proves F2 and completes the argument. □

A.5 Deriving Weight-Sites V

A.5.1 RNN

Consider the recurrent network

$$h_j(t+1) = \phi\big(Wh_j(t) + W_{\mathrm{in}}x_j(t+1)\big), \qquad h_j(0) := h_0, \tag{21}$$

with output

$$y_j(t) = W_{\mathrm{out}}h_j(t), \tag{22}$$

where W, W_{in}, and W_{out} are trainable.

The trainable weights appear only through multiplication of the current hidden state, current input, and current hidden state at readout. Thus the relevant weight-site variables are simply $h_j(t)$ and $x_j(t+1)$ for the recurrent and input weights, together with $h_j(t)$ again for the output map. If one stacks all trainable matrices into a block-diagonal matrix

$$\widetilde{W} = \mathrm{blockdiag}(W, W_{\mathrm{in}}, W_{\mathrm{out}}),$$

then the corresponding weight-site matrix may be written schematically as

$$V = \mathrm{cat}(H_-, X, H),$$

where H_- collects the pre-update hidden states $h_j(t)$, X collects the inputs $x_j(t+1)$, and H collects the hidden states feeding the readout. If the readout weights are not included in the NTK under consideration, one simply omits the final block.

In the common case where only recurrent and input weights are trained, this reduces to

$$V = \mathrm{cat}(H_-, X).$$

Consequently,

$$\mathcal{K} = VV^\top \otimes I_n,$$

up to the obvious block-size bookkeeping associated with stacking the trainable weight matrices. In expanded form, this corresponds to a sum of Gram contributions from each trainable weight family:

$$\mathcal{K} = (H_-H_-^\top + XX^\top) \otimes I_n$$

when only W and W_{in} are trained. Thus the recurrent global-state NTK is bottlenecked by the joint hidden–input activity over all batches and timesteps.

A.5.2 GRU

Consider the GRU without biases, for simplicity:

$$\begin{aligned} r_j(t+1) &= \sigma(W_{ir}x_j(t+1) + W_{hr}h_j(t)), \\ z_j(t+1) &= \sigma(W_{iz}x_j(t+1) + W_{hz}h_j(t)), \\ \ell_j(t+1) &= \tanh(W_{i\ell}x_j(t+1) + r_j(t+1) \odot (W_{h\ell}h_j(t))), \\ h_j(t+1) &= (1 - z_j(t+1)) \odot \ell_j(t+1) + z_j(t+1) \odot h_j(t). \end{aligned} \tag{23}$$

Stack the weights $W_{ir}, W_{iz}, W_{i\ell}$ into a single matrix W_{in}, and $W_{hr}, W_{hz}, W_{h\ell}$ into W. Then the model can be written in the lifted form

$$\begin{aligned} q_j(t+1) &= Wh_j(t), \\ p_j(t+1) &= W_{\text{in}}x_j(t+1), \\ h_j(t+1) &= \phi_{\text{step}}\big(q_j(t+1), p_j(t+1), h_j(t)\big), \end{aligned} \tag{24}$$

where ϕ_{step} performs the remaining GRU nonlinearities and gating, but does not involve the trainable weights explicitly.

More explicitly,

$$\phi_{\text{step}}(q, p, h) = (1 - \sigma(p_2 + q_2)) \odot \tanh\big(p_3 + \sigma(p_1 + q_1) \odot q_3\big) + \sigma(p_2 + q_2) \odot h,$$

where q_1, q_2, q_3 extract the pieces of q corresponding to $W_{hr}, W_{hz}, W_{h\ell}$, respectively, and similarly for p_1, p_2, p_3.

Thus the weight-site augmented state is simply the collection of hidden states and inputs:

$$V = \text{cat}\{(h_j(t), x_j(t+1))\}_{1 \le j \le n_x,\ 0 \le t \le n_t - 1}.$$

This is already evident from the original GRU equations: the trainable weights only multiply $h_j(t)$ and $x_j(t+1)$, and no additional weighted intermediate variables are required.

Verifying the Kronecker core. To make the decomposition explicit, define $v_j(t) = (q_j(t), p_j(t), h_j(t))$, with initial condition $v_j(0) = (0, 0, h_0)$, and let f_v denote the one-step lifted dynamics in Equation equation 24. The matrix W has shape $3n \times n$, while W_{in} has shape $3n \times n_{\text{in}}$. Then

$$D_W f_v = [h_j(t), 0]^T \otimes I_{3n}, \qquad D_{W_{\text{in}}} f_v = [0, x_j(t+1)]^T \otimes I_{3n},$$

up to the obvious reshaping induced by stacking the gates.

Let H_- denote the matrix collecting all $h_j(t)$ for $t = 0, \ldots, n_t - 1$, and let X denote the matrix collecting all $x_j(t+1)$ for $t = 0, \ldots, n_t - 1$. Then

$$\mathcal{K} = (D_W f_v)(D_W f_v)^\top + (D_{W_{\text{in}}} f_v)(D_{W_{\text{in}}} f_v)^\top = (H_- H_-^\top + XX^\top) \otimes I_{3n}.$$

Equivalently, concatenating H_- and X into the augmented state matrix V,

$$\mathcal{K} = VV^\top \otimes I_{3n},$$

which is exactly the Kronecker-core factorization used in the main text.

A.5.3 Single-Block Transformer

We write the self-attention block in the form

$$Q = X \boxed{W_Q^\top}, \qquad K = X \boxed{W_K^\top}, \qquad V_{\text{val}} = X \boxed{W_V^\top},$$

and define

$$A = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{n_{\mathrm{attn}}}}\Big)V_{\mathrm{val}}, \qquad Y_{\mathrm{attn}} = A\, W_O^\top.$$

We use V_{val} here for the value vectors to avoid notational conflict with the weight-site matrix V.

The MLP is then

$$H_{\ell+1} = \sigma\big(H_\ell\, W_\ell^\top + b_\ell\big), \qquad \ell = 1, \dots, L-1, \qquad H_1 := Y_{\mathrm{attn}},$$

with final output

$$Y = H_{L-1}\, W_L^\top + b_L.$$

The relevant shapes are

$$X \in \mathbb{R}^{B\times T\times n_{\mathrm{in}}}, \qquad A \in \mathbb{R}^{B\times T\times n_{\mathrm{attn}}}, \qquad H_\ell \in \mathbb{R}^{B\times T\times n_{\mathrm{mlp}}} \quad \text{for } \ell = 1, \dots, L-1.$$

Concatenating the trainable weights into one block-diagonal matrix

$$W = \mathrm{blockdiag}(W_Q, W_K, W_V, W_O, W_1, \dots, W_L),$$

the corresponding weight-site matrix is

$$V = \mathrm{cat}(X, X, X, A, H_1, \dots, H_{L-1}) \in \mathbb{R}^{BT\times m},$$

where

$$m = 3n_{\mathrm{in}} + n_{\mathrm{attn}} + (L-1)n_{\mathrm{mlp}}$$

is the lifted feature dimension. Therefore

$$\mathcal{K} = VV^\top \otimes I_n,$$

for the appropriate output-side dimension n induced by the stacked weight parameterization. Expanding the Gram term gives

$$\mathcal{K} = \big(3XX^\top + AA^\top + H_1H_1^\top + \cdots + H_{L-1}H_{L-1}^\top\big) \otimes I_n.$$

Hence

$$\mathrm{rank}(VV^\top) \le m = 3n_{\mathrm{in}} + n_{\mathrm{attn}} + (L-1)n_{\mathrm{mlp}}.$$

In particular, the temporal structure of the global-state NTK is bottlenecked by the joint rank of the input, attention output, and hidden activations that immediately feed trainable weights.