KPFlow: An Operator Perspective on Dynamic Collapse Under Gradient Descent Training of Recurrent Networks

James Hazelden[1], Laura Driscoll[4], Eli Shlizerman[1,2,3], Eric Shea-Brown[1,3]

[1]Department of Applied Mathematics [2]Department of Electrical & Computer Engineering [3]Computational Neuroscience Center University of Washington, [4]Allen Institute

Correspondence: jhazelde@uw.edu

Motivation

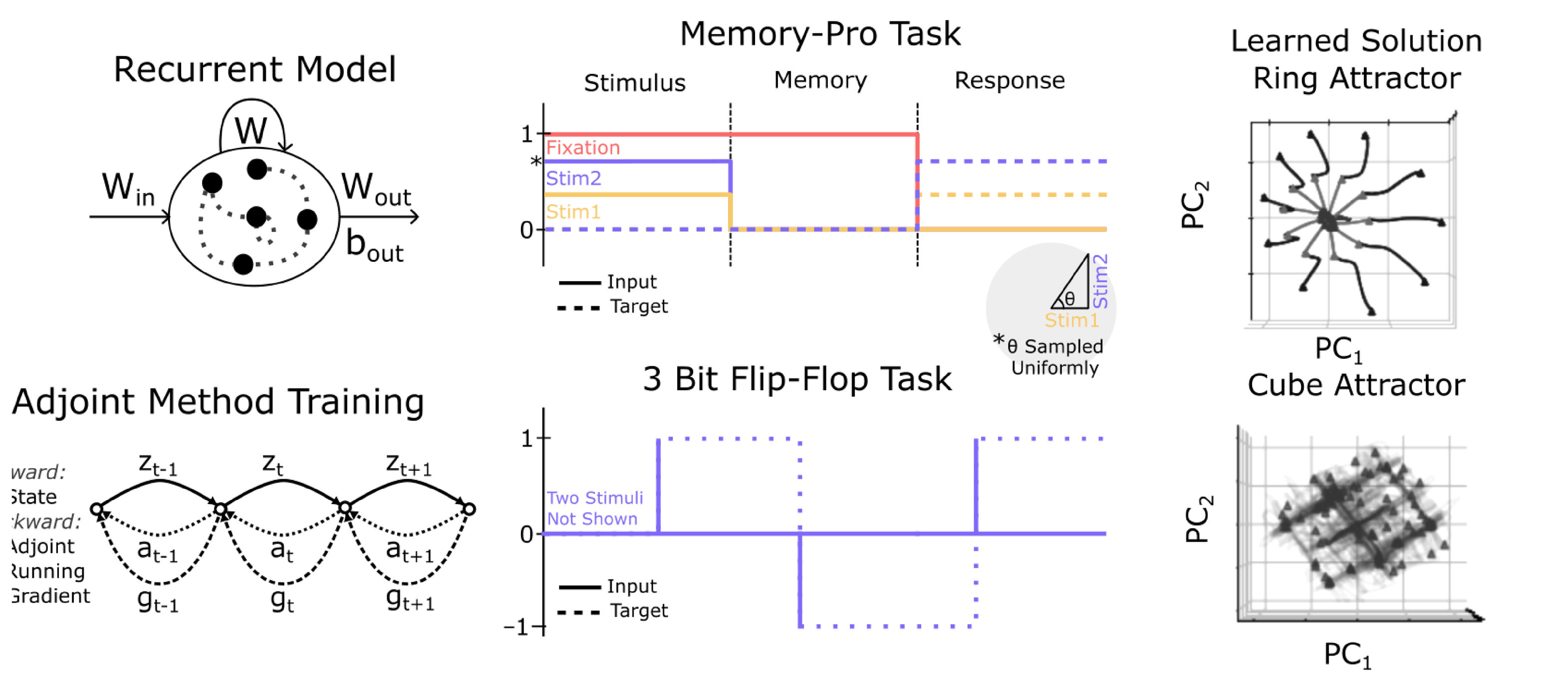


Figure 1. **Example tasks, low-dimensional collapse and learned representation.** During training on two example tasks (middle), the state collapses to a simple low dimensional motif (right panels). We develop a perturbation formula based on the adjoint that gives insights into why such collapse occurs. The adjoint approach is schematically shown in the bottom left panel.

- Recurrent dynamical systems (RNNs, GRUs, etc.) exhibit dynamical collapse to low-dimensional attractors when trained with GD [1, 2, 3]. When trained on multiple tasks, these models shared representations between each sub-task [4, 5, 6].
- **Fundamentally, there is a need for better theory to understand latent dynamics formation in general non-linear, recurrent models.**

Problem Formulation

Solve a minimization problem on parameters θ by Gradient Descent (GD):

$$\arg\min_{\theta} \left\langle \ell(z(t,x), y^*(t,x)) \right\rangle_{t,x} \tag{1}$$

where $z(t,x)$ is the hidden state on input $x \sim X$, given by a parameterized ODE:

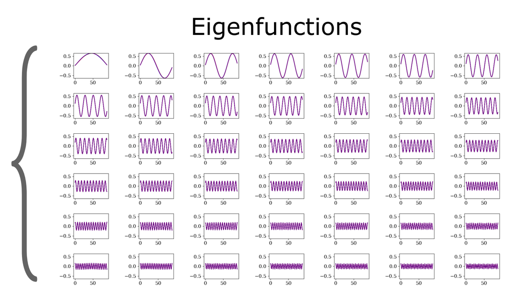
$$\frac{\mathrm{d}}{\mathrm{d}t} z(t,x) = f(z(t),x,\theta), \; z(0,x) := z_0 \tag{2}$$

Our Contributions

1. **Prop. 1 — KPFlow Decomposition** We show that gradient flow factorizes into two linear operators: $\mathcal{P}$ (the Linear Flow Propagator) and $\mathcal{K}$ (the Parameter Operator).
2. **Thm. 1 — Operator Properties** We prove the effective rank of $\mathcal{K}$ is bounded by the latent dimension of the activity for general recurrent models. $\mathcal{P}$ generalizes Lyapunov analysis to perturbations on trajectories, not individual points in time.
3. **Dynamic Collapse** $\mathcal{K}$, filtering through parameters of the model, bottle-necks dimension of updates under GD in RNNs and GRUs. Higher-rank $\mathcal{K}$ (from larger weight scales) means less dimension collapse and faster learning.
4. **Multi-Task Alignment & Interference** KPFlow decomposes linearly into interfering operators that can be used to predict which sub-tasks will share latent subspaces.
5. **Code** We efficiently implement $\mathcal{P}$ and $\mathcal{K}$ and their SVDs, providing new analysis tools.

KP-Flow Code Input

```
pop = PropagationOperator(model, inputs, hidden)
kop = ParameterOperator(model, inputs, hidden)
svs, eigfuns = (pop @ kop @ pop.T()).svd(ncomp = 32)
```



Theoretical Results

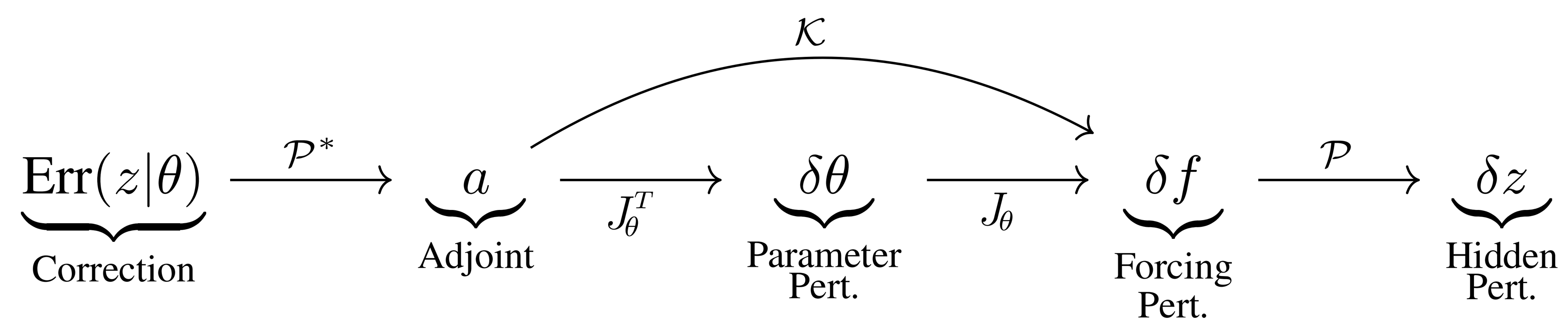


Figure 2. **Schematic of KPFlow decomposition, transforming error signals into hidden state perturbations.** Each stage of backpropagation is described by an operator on a space of 3-tensors.

Proposition 1: KP Gradient Flow Decomposition

The model dynamics in Equation 2 are perturbed by GD according to

$$\delta z = -\mathcal{P}\mathcal{K}\mathcal{P}^*(\mathrm{Err}), \quad \text{where} \;\; \mathrm{Err} := \nabla_z \ell \tag{3}$$

Where $\mathcal{P}$ and $\mathcal{K}$ are linear operators on the space of trial-dependent trajectories:

$$[\mathcal{P}\, q](t,x) \;=\; \int_0^t \Phi(t_0,t,x)\, q(t_0,x)\, dt_0 \;\; \text{where} \;\; \Phi(t_0,t) = \frac{\partial z(t,x)}{\partial z(t_0,x)} \tag{4}$$

$$[\mathcal{K}\, q](t,x) \;=\; J_\theta(t,x) \left\langle J_\theta^T\, q \right\rangle_{x_0,t_0} \;\; \text{where} \;\; J_\theta(t,x) = \frac{\partial f(t,x)}{\partial \theta} \tag{5}$$

$\mathcal{P}$: How tangential changes δf integrate into δz. Its SVD generalizes Lyapunov spectra.

$\mathcal{K}$: Filters through parameters, θ, constraining and possibly misdirecting gradient signals.

Operator SVD of $\mathcal{P}$ and $\mathcal{K}$

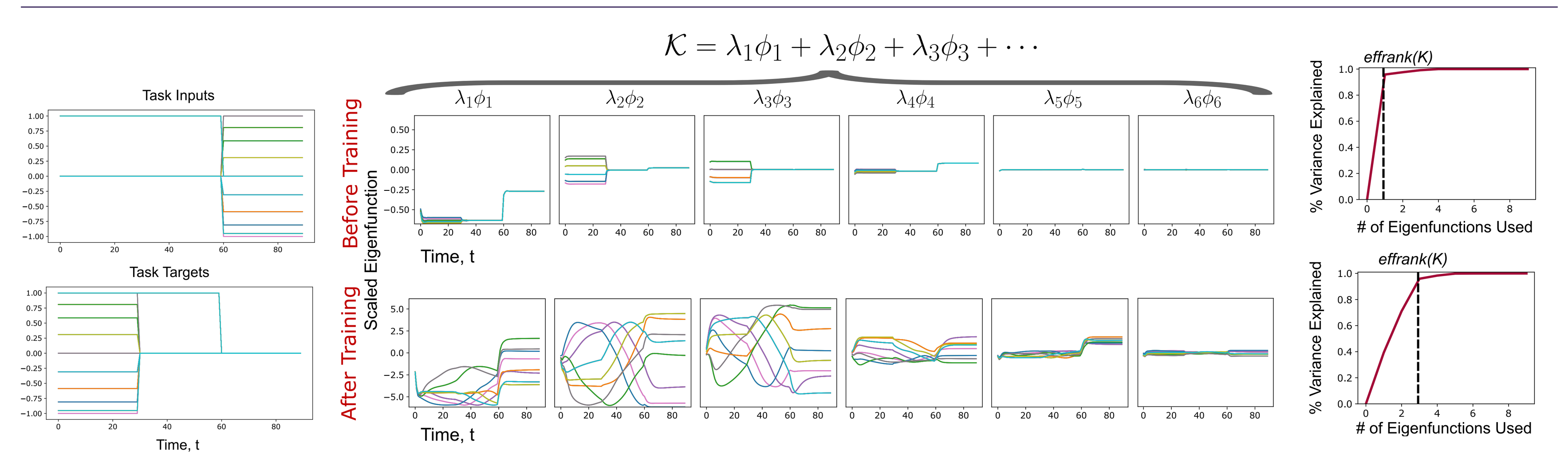


Figure 3. **Low rank eigenfunction decomposition of the $\mathcal{K}$ operator pre- and post-training on the Memory-Pro task.** Each eigenfunction is a 3-tensor over time, batch input, and hidden index.

Since $\mathcal{P}$ and $\mathcal{K}$ are linear, we can decompose them, e.g.:

$$\mathcal{P}\mathcal{P}^* = \lambda_1\phi_1 + \lambda_2\phi_2 + \ldots \tag{6}$$

For example, ϕ_1 specifies at *every* t and input x how to optimally stimulate $\mathcal{P}$.

Theorem 1: $\mathcal{K}$ Decomposes Into Simple Rank-Constrained Units

Suppose the model in Equation 2 is *weight-based*, $\theta = \{W_1, \ldots, W_M\}$, with each W_j applied once in a single evaluation of f. Then,

(1) $\mathcal{K}$ is a sum of M operators induced by each weight, $\mathcal{K} = \sum_{j=1}^{M} \mathcal{K}_j$.

(2) Each $\mathcal{K}_j$ is a positive semi-definite Hilbert-Schmidt integral operator induced.

(3) The effective rank of $\mathcal{K}_j$ over time and trials is bounded by the effective dimension of the dynamical quantity to which W_j is applied.

Neural Collapse

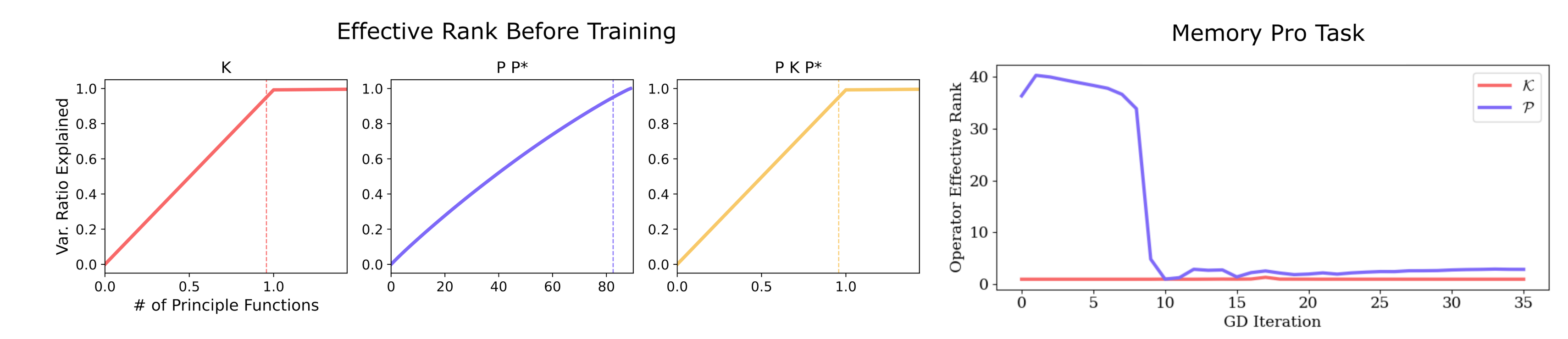


Figure 4. **$\mathcal{K}$ operator dramatically bottle-necks effective rank of learning. Left:** Cumulative explained variance ratio explained by the eigenfunctions of the operators, $\mathcal{K}$, $\mathcal{P}$ and $\mathcal{P}\mathcal{K}\mathcal{P}^*$, respectively, corresponding to an RNN at initialization with weight scale $g = 1$. **Right:** Effective rank throughout training on Memory-Pro, showing $\mathcal{K}$ is always very low rank.

- We find that for RNNs and GRUs with different non-linearities and varied initial weight scale, g, the operator $\mathcal{P}$ has effectively higher rank than $\mathcal{K}$ throughout training.
- So, $\mathcal{K}$ bottle-necks dynamical changes causing neural collapse, according to Theorem 1.

Multi-Task Latent Subspace Sharing

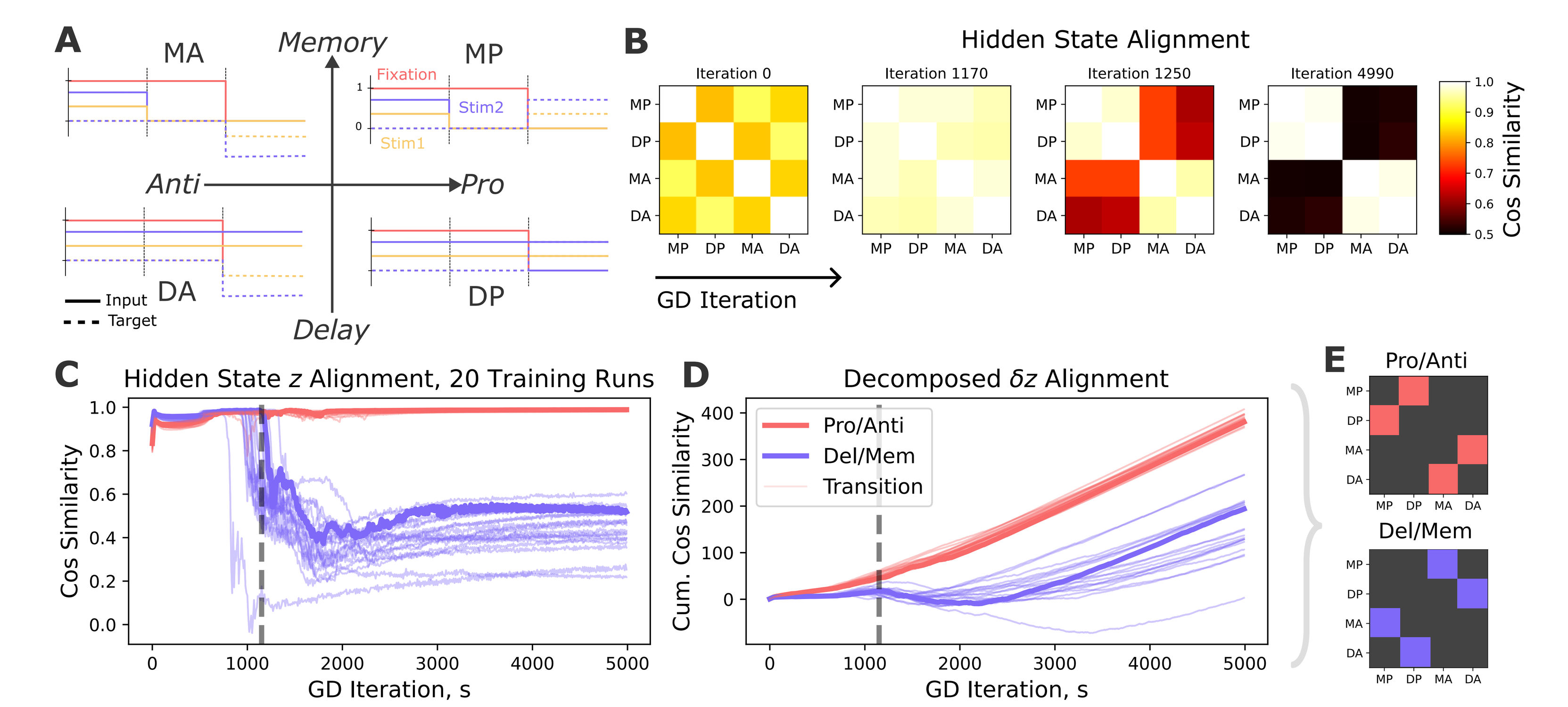


Figure 5. **Interference matrix determines emergence of subspace-aligned, shared dynamics among four tasks. A** Schematic of inputs and targets for four related tasks Memory Anti (MA), Memory Pro (MP), Delay Anti (DA), and Delay Pro (DP), on which 20 GRU were trained. **B** Cosine similarity matrices over training, measuring alignment between all hidden states. These organize into Pro/Anti configuration. **C** All 20 training runs show a transition away from Del/Mem to Pro/Anti around the same iteration. **D** Cumulative alignment based on our interference matrix. There is a preference towards Pro/Anti throughout GD, which is not visible prior to the transition in B and C.

- In a multi-task setting, $\mathcal{P}$ decomposes block-wise diagonal over inputs trials, while $\mathcal{K}$ decomposes as a linear sum:

$$\mathcal{K} = \begin{bmatrix} \mathcal{K}_{11} & \mathcal{K}_{12} \\ \mathcal{K}_{21} & \mathcal{K}_{22} \end{bmatrix}, \; \mathcal{P} = \begin{bmatrix} \mathcal{P}_1 & 0 \\ 0 & \mathcal{P}_2 \end{bmatrix}$$

- Hence, we use the operators to measure how δz corrections interact and align (Figure 4).
- This objective alignment is able to predict final shared organization, prior to them emerging under GD.

References

[1] Sussillo & Barak, *Neural Computation*, 2013 **[2]** Farrell, Recanatesi & Shea-Brown, *Current Opinion in Neurobiology*, 2023 **[3]** Mante *et al.*, *Nature*, 2013 **[4]** Driscoll, Shenoy & Sussillo, *Nature Neuroscience*, 2024 **[5]** Turner & Barak, *NeurIPS*, 2023 **[6]** Schuessler *et al.*, *NeurIPS*, 2020