Block-Recurrent Dynamics in ViTs

Mozes Jacobs[⋆a] **Thomas Fel**[⋆a] **Richard Hakim**[⋆a]
Alessandra Brondetta[b] **Demba Ba**[a,c] **T. Andy Keller**[a]

[a]Kempner Institute, Harvard University [b]University of Osnabrück [c]Harvard University

kempnerinstitute.github.io/raptor

Abstract

As Vision Transformers (ViTs) become standard backbones across vision, a mechanistic account of their computational phenomenology is now essential. Despite architectural cues that hint at dynamical structure, there is no settled framework that interprets Transformer depth as a well-characterized flow. In this work, we introduce the **Block-Recurrent Hypothesis (BRH)**, arguing that trained ViTs admit a block-recurrent depth structure such that the computation of the original L blocks can be accurately rewritten using only $k \ll L$ distinct blocks applied recurrently. Across diverse ViTs, between-layer representational similarity matrices suggest few contiguous phases. Yet, representational similarity does not necessarily translate to functional similarity. To determine whether these phases reflect genuinely reusable computation, we operationalize our hypothesis in the form of block recurrent surrogates of pretrained ViTs, which we call Recurrent Approximations to Phase-structured TransfORmers (`Raptor`). Using small-scale ViTs, we demonstrate that phase-structure metrics correlate with our ability to accurately fit `Raptor` and identify the role of training and stochastic depth in promoting the recurrent block structure. We then provide an empirical existence proof for BRH in foundation models by showing that we can train a `Raptor` model to recover 96% of DINOv2 ImageNet-1k linear probe accuracy in only 2 blocks while maintaining equivalent computational cost. To provide a mechanistic account of these observations, we leverage our hypothesis to develop a program of **Dynamical Interpretability**. We find (***i***) directional convergence into class-dependent angular basins with self-correcting trajectories under small perturbations (***ii***) token-specific dynamics, where `cls` executes sharp late reorientations while `patch` tokens exhibit strong late-stage coherence reminiscent of a mean-field effect and converge rapidly toward their mean direction and (***iii***) a collapse of the update to low rank in late depth, consistent with convergence to low-dimensional attractors.

Altogether, we find that a compact recurrent program emerges along the depth of ViTs, pointing to a low-complexity normative solution that enables these models to be studied through principled dynamical systems analysis.

1 Introduction

In the last decade, Transformers have become the default neural network architecture across machine learning communities, scaling favorably with data and compute (Vaswani et al., 2017; Kaplan et al., 2020). In particular, Vision Transformers (ViTs) (Dosovitskiy et al., 2020) have become the core architecture used in visual foundation modeling frameworks such as DINOv2 (Oquab et al., 2023; Darcet et al., 2023) and CLIP (Radford et al., 2021); and have come to dominate a wide range of visual tasks, from general visual feature extraction (He et al., 2021; Chiu et al., 2024; Yun, 2025), to diffusion (Peebles & Xie, 2023), image segmentation (Kirillov et al., 2023; Liu et al., 2024),

[⋆] Equal contribution.

Correspondence to {mozesjacobs,tfel,richhakim,takeller}@g.harvard.edu.

Preprint. Under Review.

and video processing (Arnab et al., 2021; Baldassarre et al., 2025). This increasing breadth of use motivates a move from empirical optimization to principled understanding.

Two pressures make this understanding urgent. First, safety-critical deployments (Wang & Chung, 2022; Alecu et al., 2022) demand mechanisms whose internal computation is inspectable (Losch et al., 2021), diagnosable (Adebayo et al., 2020), and verifiable (Tjeng & Tedrake, 2019) rather than opaque. As these models proliferate across domains, the ability to explain (Doshi-Velez & Kim, 2017; Gilpin et al., 2018; Kim et al., 2018), manipulate, and verify their behavior becomes increasingly essential. Second, from a scientific' inference perspective (Cichy & Kaiser, 2019), understanding what makes these models work is essential for explaining their success. Their strong performance suggests they have discovered effective strategies, and identifying these strategies could advance our broader understanding of visual intelligence. Independently of any comparison to human vision, the goal is to uncover the principles that make these systems so effective.

One promising path toward such understanding is to search for underlying simplicity. Multiple approaches explore different facets of this simplicity, whether in functional expressivity (Montúfar et al., 2014; Telgarsky, 2015; Serra et al., 2018; Balestriero et al., 2018), symmetry (Cohen & Welling, 2014; Olah et al., 2020), or computation (Wilson, 2025; Goldblum et al., 2023; Schmidhuber, 1997; Mingard et al., 2025). Discovering such simplicity principle should improve both development (Bronstein et al., 2017; Frankle & Carbin, 2019) and interpretability (Bereska & Gavves, 2024; Carvalho et al., 2019; Fel, 2024; Ghorbani et al., 2017; Fel et al., 2023; Smilkov et al., 2017; Sundararajan et al., 2017; Zeiler & Fergus, 2014; Templeton et al., 2024; Bricken et al., 2023). Depth offers a concrete place to look for this simplicity. Residual connections have long suggested a link to dynamical systems (Liao & Poggio, 2020; Veit et al., 2016; Greff et al., 2016; Boulch, 2017; Haber & Ruthotto, 2017), hinting at implicit recurrence even when layers have distinct parameters. This convergent evidence makes plausible a form of algorithmic parsimony (Ma et al., 2022) in which a small set of blocks may be reused across many layers, trading parameters for iterations. Related perspectives support this view (Dingle et al., 2020). More concretely, residual updates invite a discrete-time interpretation of depth (Chen et al., 2018; Chalvidal et al., 2020; Sander et al., 2022), attention induces coupled token dynamics (Lu et al., 2019; Geshkovski et al., 2023), and in language models contiguous block recurrence has been observed and exploited (Geiping et al., 2025; Fernando & Guitchounts, 2025; Dehghani et al., 2018; Tan et al., 2023).

More concretely, residual updates invite a discrete-time interpretation of depth (Chen et al., 2018; Chalvidal et al., 2020; Sander et al., 2022), attention induces coupled token dynamics (Lu et al., 2019; Geshkovski et al., 2023), and in language models contiguous block recurrence has been observed and exploited (Geiping et al., 2025; Fernando & Guitchounts, 2025; Dehghani et al., 2018; Tan et al., 2023). This renewed interest in recurrent computation (Jolicoeur-Martineau, 2025; Venkatraman et al., 2025) and algorithmic complexity (Shaw et al., 2025; Dingle et al., 2020) as frameworks for understanding neural network simplicity bias further motivates our investigation.

However, no existing framework characterizes depth in ViTs as representational flow or determines whether apparent phases correspond to functional reuse. Furthermore, vision explainability research (Bach et al., 2015; Fong & Vedaldi, 2017; Novello et al., 2022; Muzellec et al., 2024; Petsiuk et al., 2018; Hedström et al., 2022; Fel et al., 2025; Gorton, 2024; Kowal et al., 2024; Bau et al., 2017; Vilas et al., 2023) has not leveraged dynamical systems analysis to model emergent network structure.

In this work, we consider recurrence as a candidate source of simplicity and advance the Block-Recurrent Hypothesis (BRH): after training, the depth of a ViT organizes into a small number of contiguous phases such that the computation of the original L layers can be rewritten by reusing only $k \ll L$ distinct blocks applied recurrently. Our starting point is an empirical observation: layer–layer representational similarity matrices consistently exhibit block-diagonal structure across disparate models. However, representational similarity does not necessarily translate to functional equivalence; therefore, we ask: *Does this phase structure reflect genuinely reusable computation?*

Our contributions. Our study proceeds in three parts:

- **Empirical evidence for block-recurrent structure.** We demonstrate across diverse Vision Transformers that layer-layer representational similarity matrices exhibit distinct contiguous phases of computation, formalized through the Block-Recurrent Hypothesis. We develop a max-cut algorithm to systematically identify phase boundaries and show that this block structure (***i***) emerges during training and is (***ii***) strengthened by stochastic depth.

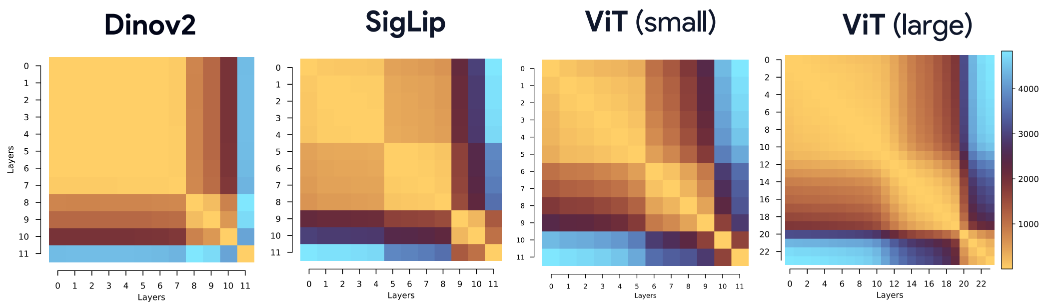

Figure 1: **Layer–layer similarity matrices across diverse Vision Transformers reveal block-structure.** Despite differences in scale and training objectives, all models exhibit contiguous block structure along depth, visible as phase-segmented regions of high similarity. Beyond representational similarity, this raises the question of whether a deeper *functional* recurrence underlies these patterns, hinting at block-wise reusability of computation across layers. In this work, we investigate this hypothesis, showing that these phase segments correspond to blocks with functional similarity, which can be approximated by a single shared block applied recurrently along depth.

- **Constructive verification via recurrent surrogates.** We operationalize the BRH by training weight-tied block-recurrent approximations of pretrained ViTs, termed `Raptor`. Critically, our goal is not compression or efficiency optimization per se, but rather to demonstrate that functional reuse is genuinely possible. `Raptor` reconstructs the complete internal representation trajectory across all layers, not merely the final output, providing strong evidence for true computational equivalence rather than input-output mimicry. Specifically, we provide empirical evidence for the BRH on foundational vision models by training a `Raptor` that recovers 96% of DINOv2's ImageNet-1k linear-probe accuracy using only 2 recurrent blocks, and 98% with 3 blocks.
- **Dynamical systems analysis framework.** Leveraging our hypothesis, we propose a program of Dynamical Interpretability that treats ViT depth as the discrete-time unfolding of an underlying dynamical system, such that the evolution of representations across layers can be analyzed as its temporal dynamics. Our analysis reveals: (***i***) directional convergence into class-dependent angular basins with self-correcting trajectories under perturbations, (***ii***) token-specific dynamics where `cls` tokens execute sharp late reorientations while patch tokens exhibit strong coherence reminiscent of mean-field behavior, and (***iii***) collapse of layer-to-layer updates to low-rank subspaces consistent with convergence to low-dimensional attractors.

As a first step, we characterize emergent phases in representation space, motivating the formulation of the Block-Recurrent Hypothesis.

2 Emergent Phase Structure & the Block-Recurrent Hypothesis

Our investigation starts with a simple experiment: we construct layer-layer similarity matrices by computing the cosine similarity of each token at layer l with the same token at layer m. As shown in Figure 1, despite significant differences in tasks, training objectives, and scale, all models exhibit consistent block-wise organization where contiguous layers exhibit high mutual similarity within blocks and lower similarity across block boundaries. This finding is not new and echoes early (and more recent) observations in residual networks (Kornblith et al., 2019; Hoang et al., 2025), but raises a fundamental question: does representational similarity reflect deeper computational structure? In fact, representational similarity alone provides no guarantee of functional equivalence. Layers might produce similar representations through entirely different computational pathways, or conversely, functionally equivalent computations might yield representations that appear dissimilar due to linear transformations or noise. The critical question is whether these apparent phases correspond to genuine functional recurrence – that is, whether the same computational operations are being reused across different layers within each phase. We formalize this possibility through the Block-Recurrent Hypothesis:

Definition 1 (Block-Recurrent Hypothesis (BRH))**.** *We consider $\boldsymbol{f}$ to be a trained Vision Transformer with nominal depth L and intermediate maps $\boldsymbol{f}_\ell : \mathcal{X} \to \mathcal{A}_\ell$, $\ell \in \{1, \dots, L\}$. We say that $\boldsymbol{f}$ satisfies the ε-BRH if for any ℓ, there exist $k \ll \ell$ blocks $\boldsymbol{B}_1, \dots, \boldsymbol{B}_k$ and integers $n_1, \dots, n_k$ with $\sum_{j=1}^{k} n_j = \ell$ such that:*

$$\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}}\big(\|\boldsymbol{f}_\ell(\boldsymbol{x}) - (\boldsymbol{B}_k^{(n_k)} \circ \cdots \circ \boldsymbol{B}_1^{(n_1)})(\boldsymbol{x}))\|_F\big) \le \varepsilon,$$

where $\boldsymbol{B}_j^{(n_j)}$ denotes n_j repeated applications of the same parameter-tied block $\boldsymbol{B}_j$ and the entire approximation maintaining equivalent computational cost.

Here, $||\cdot||_F$ denotes the Frobenius norm and $\mathbb{P}$ is a probability distribution over natural images. To put it simply, this hypothesis state that a ViT's L layers can be replaced by $k \ll L$ recurrent blocks that reproduce the entire internal trajectory at equivalent computational cost. By requiring intermediate layer fidelity rather than just final output matching, we rule out trivial solutions where computation is concentrated in a single block. The constraint $k \ll L$ with parameter tying ensures genuine functional reuse rather than simple parameter copying.

To test this hypothesis, the first step is to operationalize it by proposing a method for constructing such approximations. We naturally turn to recurrent architectures but develop a specialized training technique that we describe now.

Operationalizing Block-Recurrence with `Raptor`. Since the BRH asserts only the existence of recurrent blocks satisfying its conditions without specifying their precise form, the most direct validation is constructive: demonstrating existence by example of a recurrent model approximating $\boldsymbol{f}$.

However, recurrent architectures are notoriously difficult to train due to error accumulation and instability: small prediction errors compound autoregressively across depth, leading to exploding or vanishing gradients and divergent trajectories (Pascanu et al., 2013; Linsley et al., 2020; De et al., 2024). Standard backpropagation through time becomes unstable as recurrence depth increases, and the model must learn to simultaneously handle both the forward dynamics and its own prediction errors in closed loop (Williams & Zipser, 1989). To circumvent these challenges, we will leverage the intermediate layer activations as training targets, enabling a staged approach that combines the stability of teacher forcing with the self-consistency required for autoregressive deployment. More precisely, we introduce a procedure to distill existing Vision Transformers into Recurrent Approximations to Phase-structured TransfORmers (`Raptors`), using k parameter-tied blocks with repetition counts determined by a max-cut phase discovery algorithm. This approach transforms the abstract hypothesis into a concrete architectural and training framework that can be empirically validated.

This constructive approach requires that `Raptor` models reproduce the internal activations of the full ViT they approximate, similar to Dasgupta & Cohn (2025); Sanh et al. (2019); Shleifer & Rush (2020), not merely mimic the final output[1]. The BRH implies that such reproduction should be possible within tolerance ε, making activation matching a natural training objective. Formally, let $\boldsymbol{f}$ be a reference ViT with intermediate activations $\boldsymbol{a}_\ell(\boldsymbol{x}) \equiv \boldsymbol{f}_\ell(\boldsymbol{x}) \in \mathbb{R}^{t\times d}$ for $\ell = 0, \ldots, L$, where layer $\ell = 0$ denotes the patch encoder and $1 \leq \ell \leq L$ refer to transformer layers. Here, t is the number of tokens and d the feature dimension. Let $\boldsymbol{B}_j$ denote the j-th parameter-tied block in our recurrent decomposition. The `Raptor` approximation produces activations:

$$\tilde{\boldsymbol{a}}_\ell(\boldsymbol{x}) \equiv (\boldsymbol{B}_k^{(n_k)} \circ \cdots \circ \boldsymbol{B}_1^{(n_1)})(\boldsymbol{a}_0(\boldsymbol{x})) \tag{1}$$

where the composition covers layers 1 to ℓ according to our phase segmentation. We train `Raptor` using an autoregressive loss (AR) that enforces trajectory fidelity across all intermediate layers:

$$\mathcal{L}_h^{\text{AR}}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{x}}\big(\sum_{\ell=1}^{h} \|\tilde{\boldsymbol{a}}_\ell(\boldsymbol{x}) - \boldsymbol{a}_\ell(\boldsymbol{x})\|_F\big), \quad h \leq L. \tag{2}$$

With this approach, each block learns to approximate its designated contiguous segment while the overall model reproduces the complete representational trajectory of the teacher network. However, this formulation does not specify how to determine the block boundaries or phase assignments. We address this now with a simpler algorithmic approach based on the representational similarity structure observed earlier.

Choosing partitions. For a given number of blocks k, we must introduce a practical method to determine the number of recurrent iterations of each block (n_k); in other words, where the recurrent "phases" of computation begin and end. We accomplish this by casting this "block discovery" process

[1] Unlike classical distillation, which typically supervises logits (and occasionally a few intermediate "hints"), we enforce one-to-one alignment of all layers representations across the entire depth for the same inputs. The recurrent surrogate must generate the teacher's intermediate activations, not just its predictions.

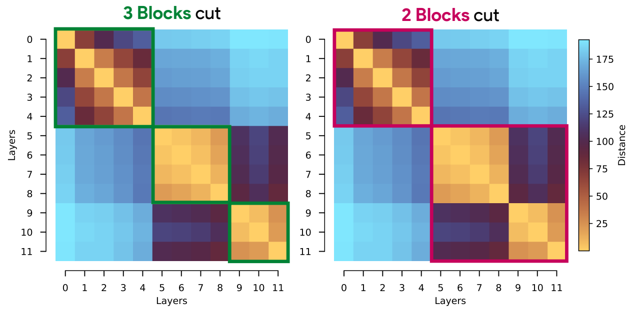

Figure 2: **Block discovery via max-cut segmentation of the layer–layer similarity matrix.** Our algorithm partitions depth into contiguous segments by maximizing within-block similarity and minimizing cross-block cosine similarity. Shown are two cuts of the same ViT-B: with 3-blocks (left, green) and 2-blocks (right, magenta). These cuts reveal candidate block boundaries where the representation dynamics undergo sharp transitions, providing an operational method for detecting recurrent phases in trained Vision Transformers.

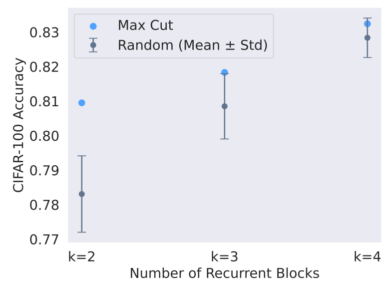

Figure 3: **Evaluation of `Raptor` models on CIFAR-100 using our max-cut partitioning algorithm versus random contiguous partitions.** Reported values are classification accuracy. Results for random partitions are aggregated over 10 different random partitions.

as a weighted max-cut problem Goemans & Williamson (1995), solved via dynamic programming (see Subsection B.1 for details). Specifically, the algorithm seeks to partition depth into contiguous segments by maximizing within-block similarity and minimizing cross-block similarity. We visualize the results of this procedure applied to ViT-B in Figure 2, demonstrating that the discovered blocks align reasonably with qualitative assessment.

To validate this approach, we train recurrent transformer models using max-cut partitions to reproduce the activations of trained vision transformers on CIFAR-100 (validation set accuracy 90.7%). Remarkably, as shown in Figure 3, `Raptor`s with only 2 recurrent blocks closely match the performance of the full models they approximate. The max-cut algorithm provides partitions that achieve strong performance out of the box, with accuracy near or even exceeding one standard deviation of randomly chosen partitions. This initial success on smaller ViTs provides compelling evidence that the block-recurrent structure is not merely representational but genuinely functional.

How do blocks emerge? Having operationalized the BRH and demonstrated a method for block discovery, we now turn to the origin of such principle: *under what conditions does this block-recurrent structure emerge in trained Vision Transformers* ? To investigate this systematically, we examine small-scale ViTs where we control training conditions and isolate potential contributing factors. Specifically, we hypothesize that training and stochastic depth (Huang et al., 2016) may promote the emergence of block-recurrent patterns.

Motivated by evidence that residual networks tolerate variable effective depth (Wu et al., 2019), we examined the effect of stochastic depth (SD) on block recurrence. During training, each layer is independently dropped with probability p, applied uniformly across depth. We trained ViT-B/14 from random initialization on CIFAR-100, using the `cls` token for the linear probe across a sweep of SD p rates. We observe an increase in layer-layer similarity with increasing SD p rates (Figure 4A). We next used these trained ViT networks as teachers for student `Raptor` models (see Appendix B). `Raptor` models were trained to reconstruct the hidden activation states of the teacher ViT across layers. `Raptor` forward passes are fully autoregressive, meaning each layer's output is fed into the next layer and is also trained to match the corresponding layer in the teacher network. We quantify the similarity of the `cls` and patch token representations in each layer between the teacher and student networks as the R^2 of their matched token embeddings (Figure 4B).

We observe that, as stochastic depth increases, a separately trained `Raptor` model becomes significantly better at reconstructing the ViT's internal hidden states. These results demonstrate that stochastic depth regularizes the ViT to learn a representational trajectory that is more compressible into a recurrent form. The observed decoupling between student accuracy and teacher–student reconstruction fidelity (Figure 4C) suggests that recurrence-based compressibility does not require a trained teacher. Indeed, we observe that `Raptor` can also reconstruct hidden states of a randomly initialized ViT (Figure 12). [Fel]: We need 2 more baseline here and fix the text accordingly (plus abstract and contribution at the end of intro)

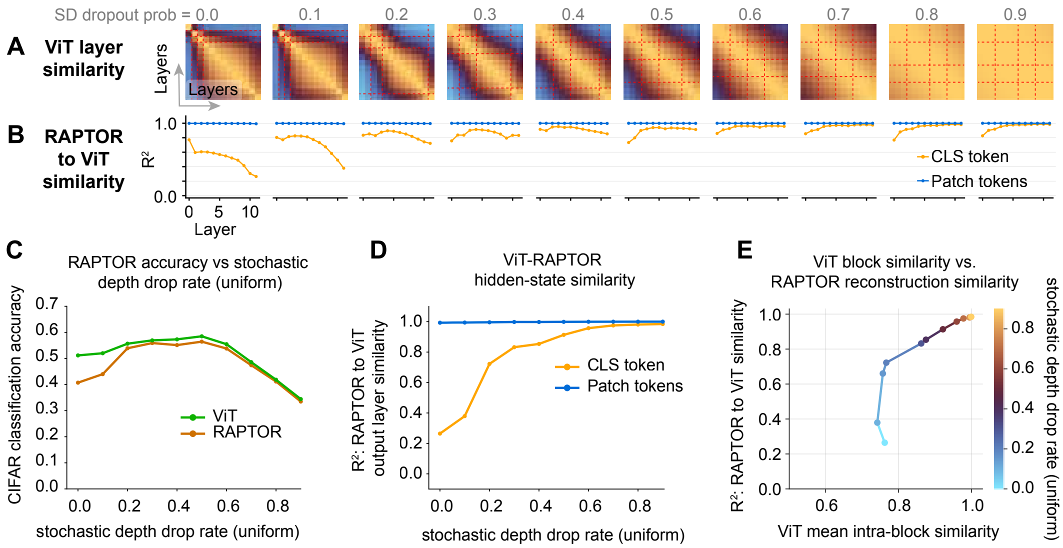

Figure 4: **Stochastic depth promotes representational similarity across layers block-recurrence. A)** ViT layer-layer cosine similarity matrices for models trained with increasing stochastic depth (SD) dropout probability p (probabilities of 0.0-0.9, uniform over layer depth). Dashed red lines delineate blocks, as defined by the max-cut algorithm. Higher SD p values lead to a more similar representation across layers. **B)** Layerwise teacher-student representational alignment R^2 (`Raptor` vs. ViT) of the class `cls` and patch tokens. Increases in SD p correspond to an increase in the ability of `Raptor` to match the ViT's layerwise representations. **C)** CIFAR classification accuracy for a ViT trained on CIFAR, and a `Raptor` model with $k = 3$ blocks trained to match the hidden state of the ViT. The gap in performance between the teacher ViT and student `Raptor` models narrows as the SD p rate grows (applied to the ViT training). **D)** Last layer hidden-state similarity R^2 of the ViT and `Raptor` model as a function of SD p. Increases in stochastic depth lead to a greater ability to reconstruct ViT function using `Raptor` . **E)** Association between layer-layer representational similarity and `Raptor` reconstruction R^2. Stochastic depth encourages the formation of more similar blocks of layers within the ViT, which facilitates approximation by the recurrent `Raptor` model.

We further quantified this relationship, and observed that while ViT image classification performance peaks with an intermediate SD probability (Figure 4C), teacher-student ViT-`Raptor` representation reconstruction consistently improves with increasing SD probability (Figure 4D). We combine the above results in Figure 4E and observe a strong positive association between the ViT's layer-layer representational similarity and `Raptor` reconstruction fidelity. These results support the view that the representational block structure seen in small-scale and foundation models reflects an emergent functional recurrence that can be quantified and exploited via architectural recurrence. Using the methods established here, we next scale up our application of `Raptor` to modern large-scale foundation models.

3 Scaling `Raptor` to Foundation Models

Having demonstrated the BRH on controlled experiments, we now test whether it extends to large-scale foundation models. We apply `Raptor` to DINOv2, chosen for its widespread adoption across vision tasks, and optimize it to reproduce DINOv2's internal activations on ImageNet-1k.

Architecture and Training. For all experiments, we use ImageNet-1K and extract activations from a pretrained DINOv2 (ViT-Base) model, applying our max-cut algorithm to identify $k \in \{2, 3, 4\}$ recurrent block partitions. Each recurrent block $\boldsymbol{B}(\cdot)$ mirrors the block of DINOv2 (for detail about the block see Section B).

For the training of the block, an attentive reader will notice that the autoregressive formulation, while conceptually elegant, presents a scalability bottleneck: training requires sequential application of blocks using self-generated predictions. However, a simple modification unlocks massive parallelization: we can train blocks to predict the next layer given *ground-truth activations* from the pretrained model, rather than their own predictions. This teacher forcing (TF) approach is naturally parallel: all training examples and all blocks can be processed concurrently without sequential dependencies.

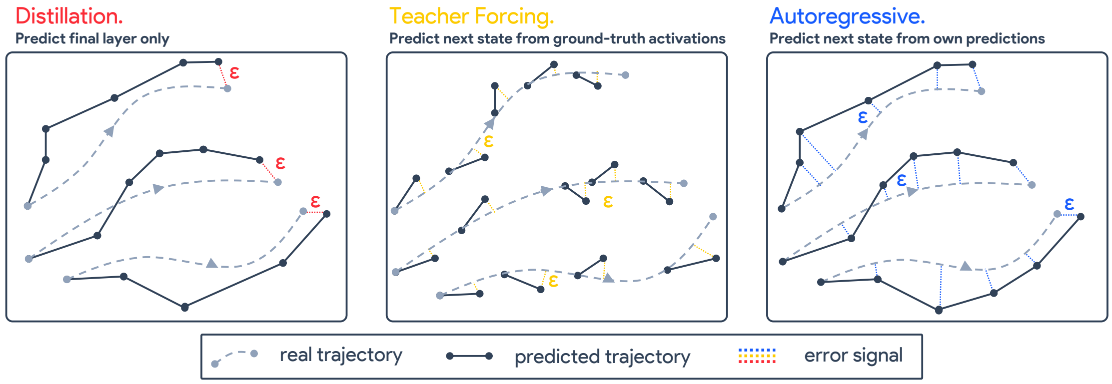

Figure 5: **Three training paradigms for learning recurrent approximations.** Each panel shows three token trajectories through depth. Gray dashed lines with filled circles represent the ground-truth teacher trajectories; black solid lines with filled circles show the student's predictions; colored dotted lines (with ε labels) indicate the error signal between predicted and ground-truth states. **Left (Distillation):** The student network directly predicts the final layer from the initial state, with no supervision on intermediate representations. Error is measured only at the terminal state, providing no guidance on the representational trajectory. **Middle (Teacher Forcing):** At each depth step ℓ, the student block predicts $\hat{\boldsymbol{x}}_{\ell+1} = \boldsymbol{B}(\boldsymbol{x}_\ell)$ using the ground-truth activation $\boldsymbol{x}_\ell$ from the teacher. Vertical arrows indicate where the student "resets" to ground-truth states. This enables efficient parallel training and prevents error accumulation, but creates a train-test mismatch since the model never learns to handle its own prediction errors. **Right (Autoregressive):** The student autoregressively predicts $\hat{\boldsymbol{x}}_{\ell+1} = \boldsymbol{B}(\hat{\boldsymbol{x}}_\ell)$ using its own previous predictions, matching inference conditions. Errors can compound across depth (shown by increasing deviation between trajectories), requiring the model to learn self-consistent, closed-loop dynamics. Our two-stage training (Sec. 3) combines both approaches: Stage 1 uses teacher forcing for stable, parallelizable pretraining; Stage 2 switches to autoregressive training to ensure self-consistency at inference.

Formally,

$$\text{Teacher Forcing:} \quad \begin{cases} \hat{\boldsymbol{a}}_{\ell+1}(\boldsymbol{x}) \equiv \boldsymbol{B}_k^{(1)}(\boldsymbol{a}_\ell(\boldsymbol{x})), \\ \mathcal{L}^{\text{TF}}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{x}}\big(\sum_{\ell=0}^{L-1} \|\hat{\boldsymbol{a}}_{\ell+1}(\boldsymbol{x}) - \boldsymbol{a}_{\ell+1}(\boldsymbol{x})\|^2\big). \end{cases} \tag{3}$$

We thus propose a 2-stage training, a pre-training stage and a finetuning state. Stage 1 (TF pretraining) leverages this parallelization for efficient large-scale training, with each block learning to predict next-layer activations from real intermediate features. Stage 2 (AR finetuning) switches to the autoregressive objective where blocks consume their own predictions, ensuring self-consistency at inference time. This combines the computational efficiency of parallel teacher forcing with the autoregressive reliability required for downstream performance, which yields the following total loss:

$$\mathcal{L}_{\text{total}}(\boldsymbol{x}) = \underbrace{\lambda\, \mathcal{L}^{\text{TF}}(\boldsymbol{x})}_{\text{Pretraining: } \lambda=1} + \underbrace{(1-\lambda)\, \mathcal{L}_h^{\text{AR}}(\boldsymbol{x})}_{\text{Finetuning: } \lambda=0} + \Omega(\boldsymbol{\theta}),$$

where $\Omega(\boldsymbol{\theta})$ denotes additional regularization applied to each tied block. See appendix B for complete details. To summarize, teacher forcing enables parallel training at the cost of using ground-truth rather than self-generated activations. Stage 1 exploits this for efficient pretraining (blocks develop their computational roles simultaneously on real activations). Stage 2 then connects blocks into the complete recurrent architecture, switching to pure autoregressive training where blocks must coordinate and handle their own predictions. This transition from parallel efficiency to autoregressive consistency ensures the final model operates independently while preserving the original network's representational trajectory. We provide an implementation framework at `https://kempnerinstitute.github.io/raptor`.

With the training methodology established, we now evaluate how effectively `Raptors` can reproduce the performance of their teacher networks across multiple vision tasks.

Results. We evaluate `Raptor` against DINOv2 by training linear probes on ImageNet-1k (classification), ADE20k (semantic segmentation), and NYUv2 (monocular depth), covering both classification and dense prediction. For ImageNet-1k, we initialize the classifier from the public DINOv2 probe

Method	Arch.	IN-1k (Acc ↑)	ADE20k (mIoU ↑)	NYUv2 (RMSE ↓)
`Raptor`	$k = 2$	81.3	39.8	0.636
	$k = 3$	83.0	43.1	0.621
	$k = 4$	83.1	43.5	0.618
DINOv2	ViT-S	81.1	44.6	0.598
	ViT-B	84.6	47.6	0.577

Table 1: **Performance of `Raptor` compared to DINOv2 with linear probes.** We report top-1 accuracy on ImageNet-1k, mean Intersection-over-Union (mIoU) on ADE20k semantic segmentation, and root mean squared error (RMSE) on NYUv2 depth estimation. Higher values are better for accuracy and mIoU, while lower values are better for RMSE. For `Raptor`, *Arch* denotes the number of recurrent blocks, while for DINOv2, *Arch* denotes the vision transformer backbone.

and report the best score across initialization and fine-tuning. In all experiments, the ViT backbone is frozen for both `Raptor` and DINOv2; only the linear heads are updated, and we reuse DINOv2's final layer normalization (also frozen). Results appear in Table 1. `Raptor` performs well across tasks and is stronger on classification: with $k = 3$ it attains 83.0% top-1 on ImageNet-1k (about 98% of DINOv2 ViT-B and above ViT-S; see Fig. 6).

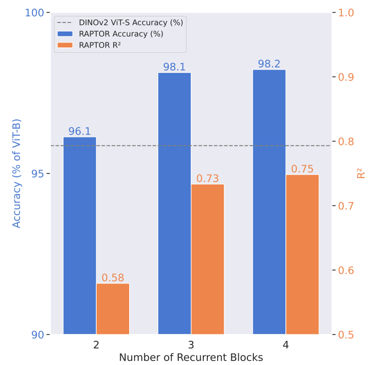

Figure 6: **`Raptor`'s performance on ImageNet-1k as a function of DINOv2 ViT-B accuracy (left), and R^2 score (right).** DINOv2 ViT-S accuracy shown as a dashed horizontal line.

Accuracy improves markedly from $k = 2$ to $k = 3$ and then saturates at $k = 4$. In short, a two-block `Raptor` at iso-FLOPs retains about 96% of DINOv2 ViT-B with a frozen backbone, a compact rewriting that substantiates the BRH.

Ablations. Although our aim is not maximal compression nor exact accuracy matching, we perform targeted ablations to identify the factors most critical to `Raptor` performance (Table 2).

Training with teacher forcing alone (Stage 1 only), while computationally efficient, leads to complete collapse with worse-than-random accuracy $\sim 1\%$ on ImageNet-1K), indicating that one-step supervision is insufficient without exposure to the full autoregressive trajectory. Introducing the autoregressive loss and gradually annealing teacher forcing to zero raises accuracy by more than 68%, underscoring the necessity of closed-loop training for stable block-recurrent approximation. Further gains come from depth scaling, where each block has a learned scalar embedding of its target layer index, making `Raptor` a non-autonomous dynamical system (i.e., the update rule explicitly depends on the iteration count rather than state alone). Additional improvements come from up-weighting the `cls` token loss in the final block (see Eq. 4, Appendix B). Finally, connecting all blocks and fine-tuning the model end-to-end with the autoregressive objective produces a dramatic jump in performance, and a final boost is obtained by fine-tuning the linear probe. Now that we have shown that the BRH holds for a foundation model, and before turning to Dynamical Interpretability, we first examine one implication of this phenomena: the algorithmic and computational implications of the BRH.

Method	Accuracy
Teacher Forcing (TF)	3.9
+ Autoreg (anneal TF)	72.7 ↑ 68.8
+ Depth Scaling	75.2 ↑ 2.5
+ Weighted `cls`	76.7 ↑ 1.5
+ Connect	82.4 ↑ 5.7
+ Finetune (Classifier)	83.0 ↑ 0.6

Table 2: **Ablations to original (TF) algorithm with `Raptor(k=3)`**, showing ImageNet-1k accuracy with DINOv2 pretrained linear classifier. Connect refers to putting all three blocks together and training the full model autoregressively.

Algorithmic and computational implications. At scale, BRH holds in practice: a two-block `Raptor` recovers most of DINOv2 ViT-B, with three blocks essentially closing the gap. This reveals a strong simplicity bias in trained ViTs: depth reuses a small set of computations, effectively trading parameters for iterations.

This reuse has two immediate consequences. First, it compresses the program description length that realizes the network's computation, suggesting low algorithmic complexity. However, the implication is more subtil than standard Kolmogorov complexity (Kolmogorov, 1965). While Kolmogorov compression can replace a long program with an arbitrarily short one that runs in unbounded time, `Raptor` crucially preserves computational cost: applying the same block n_j times achieves equivalent run-

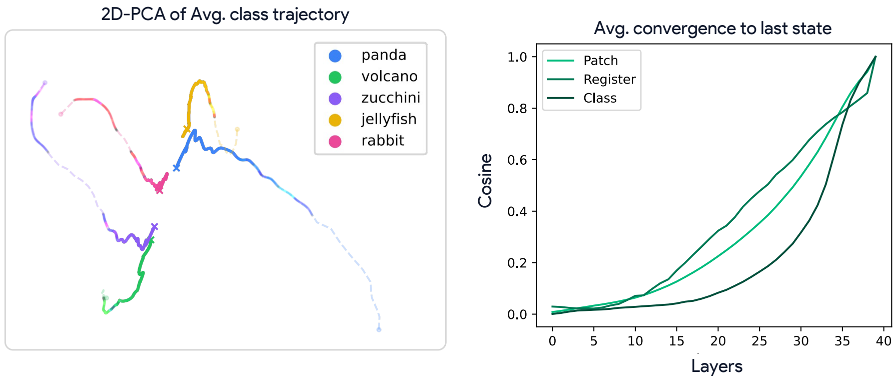

Figure 7: **Directional convergence on the unit sphere. (Left)** Qualitative view of the average normalized trajectories (in PCA space) shows collapse into compact class-dependent basins, consistent with low-dimensional angular attractors. **(Right)** Quantitative measure of cosine to final token representation γ_ℓ are S-shaped and saturate near 1 for `cls`, `registers`, and `patch`, indicating directional fixed points.

time to n_j distinct untied blocks. In other words, ViTs admit a more compact program representation *under the same runtime budget*, aligning more closely with Levin's complexity K_{Levin} (Levin, 1973).

Proposition 1 (BRH guarantees low Levin complexity)**.** *Let $\boldsymbol{f}_\ell$ satisfy 0-BRH with $k \ll \ell$ tied blocks $\{\boldsymbol{B}_j\}_{j=1}^k$ and schedule $(n_j)_{j=1}^k$ with $\sum_j n_j = \ell$. Let $R(\cdot)$ denote block runtime and define the untied teacher runtime $R(\boldsymbol{f}_\ell) := \sum_{j=1}^k n_j R(\boldsymbol{B_f})$; assume runtime parity $R(\boldsymbol{B}_j) \leq (1+\delta)R(\boldsymbol{B_f})$. Then*

$$K_{\text{Levin}}(\boldsymbol{f}_\ell) \leq \sum_{j=1}^{k} DL(\boldsymbol{\theta}(\boldsymbol{B}_j)) + O(k \log \ell) + \log R(\boldsymbol{f}_\ell) + O(1).$$

See Appendix D for details. Theoretically, BRH guarantee low Levin complexity (compact algorithmic descriptions at unchanged computational cost). Our empirical validation of BRH on DINOv2 then suggest that foundation vision models are algorithmically simpler than their nominal architecture suggests. This compression reinforces emerging evidence that simplicity principles govern successful neural networks (Goldblum et al., 2023; Valle-Perez et al., 2019; Huh et al., 2023). For interpretability research, this offers an encouraging perspective: there exist representational lenses under which seemingly complex models reveal underlying simplicity. High-performing ViTs discover and iteratively reuse a compact set of algorithmic primitives, which is a structural regularity that may provide tractable entry points for mechanistic understanding.

We now pursue one such entry point. Since ViTs compress to recurrent blocks applied iteratively, we propose to analyze their computation as discrete-time dynamical systems, in the next section, we develop this *Dynamical Interpretability* framework to extract insights from the recurrent structure.

4 From Block Recurrence to *Dynamical Interpretability* in VITs

Having observed block-structured representational similarity and confirmed that this similarity translates to functional recurrence, even in foundational models, we are naturally inclined to now seriously consider Vision Transformers as dynamical systems that can be interpreted using dynamical systems analysis tools – what we term *dynamical interpretability*. We begin by establishing the basic dynamical properties of this depth flow, and present three key findings: (***i***) tokens converge directionally toward angular attractors with self-correcting dynamics, (***ii***) different token types exhibit specialized dynamics with punctuated transitions at phase boundaries, and (***iii***) later layers exhibit low-rank collective motion under weak contraction, reminiscent of mean-field processes with collapsing update dimensionality.

Directional Convergence and Angular Attractor Geometry.
We begin by isolating direction and scale. Feature norms increase steadily with depth across token types, which precludes any Euclidean notion of convergence or attractors; we therefore normalize states and study their angular evolution (Fig. 8). Concretely, let $\hat{\boldsymbol{x}}_\ell = \boldsymbol{x}_\ell / \|\boldsymbol{x}_\ell\|$ denote the direction of a token at layer ℓ, and consider the depth trajectory $\{\hat{\boldsymbol{x}}_\ell\}_{\ell=0}^L$ on the unit sphere $\mathbb{S}^{d-1}$.

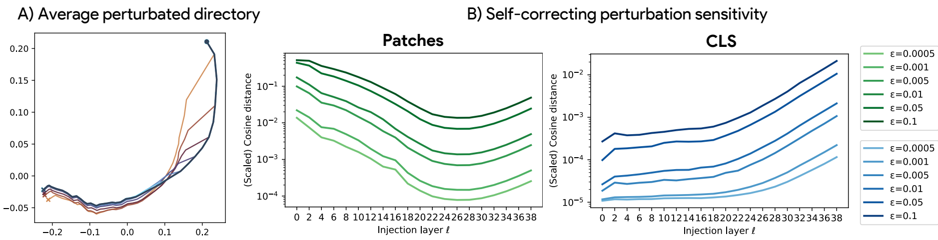

Figure 9: **Self-correction under small angular perturbations.** Perturbed trajectories bend back toward the baseline path, evidencing local basin stability. Sensitivity decays approximately log-linearly with remaining depth for `patch` tokens, but grows late for `cls`, consistent with stronger late-stage aggregation.

Directional convergence is quantified by $\gamma_\ell = \langle \hat{\boldsymbol{x}}_\ell, \hat{\boldsymbol{x}}_L \rangle$. Empirically, γ_ℓ follows smooth S-shaped curves that approach 1 and saturate in late layers for all token types (Fig. 7, right). This behavior indicates a directional fixed point: while norms may continue to grow, directions stabilize so that $\hat{\boldsymbol{x}}_{\ell+1} \approx \hat{\boldsymbol{x}}_\ell$ as ℓ increases. The acceleration of γ_ℓ near the end of depth suggests phase-local attraction that strengthens in the final phase, we clarify this with our coherence study (Figure 11, middle). A complementary geometric view comes from projecting the depth trajectories onto a low-dimensional subspace. PCA reveals that sample-specific paths enter class-dependent basins in a shared angular subspace, we took $1,000$ images coming from 5 imagenet classes with trajectories curling into compact terminal regions rather than scattering (Fig. 7, left). We interpret these regions as angular attractors: small sets on $\mathbb{S}^{d-1}$ toward which iterates of the phase-local map steer directions, up to within-class variability. Finally, we probe stability by injecting a small additive perturbation at layer ℓ and following the perturbed direction thereafter. The average perturbed path bends back toward the unperturbed trajectory, indicating local self-correction and on-sphere contraction around the limiting direction (Fig. 9). Taken together, these measurements establish property (i): token directions evolve under depth toward angular attractors with mild contraction, making directional geometry an appropriate lens for subsequent dynamical analysis.

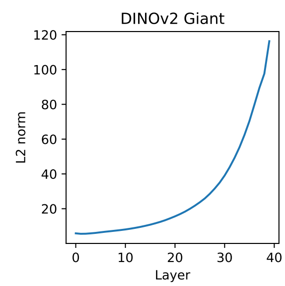

Figure 8: **Depth-wise feature norms.** Magnitudes grow with depth, motivating analysis on directions.

Token-Specific Dynamics. Token groups follow distinct angular update laws. For a token with normalized state $\hat{\boldsymbol{x}}_\ell$ define the per-layer angular speed $s_\ell = \arccos\langle \hat{\boldsymbol{x}}_{\ell+1}, \hat{\boldsymbol{x}}_\ell \rangle$. Aggregating s_ℓ by token type reveals stable small speeds for registers, intermediate speeds for patches, and sharp late reorientations for `cls` (Fig. 10). The variance of s_ℓ is smallest for registers after early depth, by contrast, `cls` exhibits increased angular activity near the end, consistent with its function as a global aggregator. These token-specific laws are not uniform across depth. Angular speed statistics display abrupt changes aligned with previously discovered block boundaries, producing a punctuated pattern in which each phase maintains near-stationary behavior that is reset at phase transitions (Fig. 10). This structure matches the block-recurrent view in which a phase applies a reused update map with stable statistics before handing off to a new regime at the boundary. Sensitivity analyses corroborate this specialization. Inject a small additive perturbation of magnitude ε at layer ℓ and measure the final angular deviation using the cosine distance $d_{\cos}(\hat{\boldsymbol{x}}_L^{(\varepsilon,\ell)}, \hat{\boldsymbol{x}}_L)$. The scaled sensitivity $|\varepsilon|^{-1} d_{\cos}$ decays approximately log-linearly with deeper injection for patch tokens, indicating on-sphere attenuation within phases, whereas it increases for `cls` when injected late, indicating accumulation at the readout stage where global information is consolidated (Fig. 9B). Directional convergence rates mirror these roles. When tracking $c_\ell = \langle \hat{\boldsymbol{x}}_\ell, \hat{\boldsymbol{x}}_L \rangle$ by token type, registers approach their terminal directions earliest, patches follow with a smoother rise, and `cls` saturates only in the final phase where its reorientation peaks (Fig. 7, left). Together, these measurements show that ViT depth implements specialized, phase-local dynamics with phase transitions, consistent with block-recurrent computation.

Low-Rank Collective Motion and Linearized Depth Flow. We quantify the dimensional structure of layer-to-layer updates and observe a progressive collapse to a low-dimensional regime. For token-wise angular updates (see App. C). Both the stable rank and effective rank decrease steadily with depth, reaching values near six in the final phase, indicating confinement to a restricted subspace

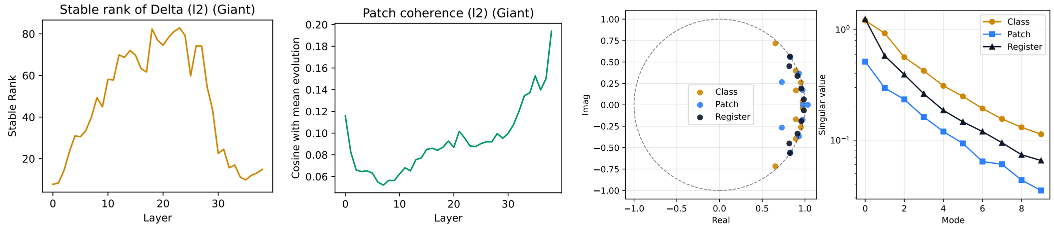

Figure 11: **(Left) Low-rank updates and coordinated patch motion.** Left Stable and effective rank of the layer-to-layer update matrix collapse with depth, indicating confinement to a restricted subspace. Right Patch-token coherence with their mean update direction rises strongly, revealing increasing collective alignment. **(Right) Dynamic Mode Decomposition (DMD) of depth dynamics.** For each token group (`cls`, registers, `patch`), we average token states within the group and fit the exact-DMD (see Section D). Each layer state is ℓ_2-normalized to unit norm (trajectories on the unit sphere), so eigenvalue angles $\arg(\lambda_i)$ characterize angular updates, while radii $|\lambda_i|$ measure contraction on the sphere (not absolute feature-norm growth). The DMD eigenvalues $\{\lambda_i\}$ lie just inside the unit circle (dashed) and concentrate near the positive real axis, indicating near-neutral, mostly angular updates with mild on-sphere contraction. `cls` modes lie closest to $+1$ (longest memory), registers are slightly more dispersed, and `patch` shows the widest angular spread and stronger contraction. The `cls` spectrum decays slowest (highest effective rank/complexity), registers are intermediate, and `patch` decays fastest (lower-rank dynamics). Together, these spectra support a weakly contracting, block-recurrent depth flow with token-specific complexity.

(Fig. 11, left). In parallel, the patch-token coherence κ_ℓ rises sharply and peaks late, showing increasingly aligned, collective updates (Fig. 11, middle). The joint pattern (rank collapse with rising coherence) marks a transition from many weakly independent directions to a few shared directions. We then linearize the depth flow via exact DMD on group-averaged, ℓ_2-normalized states, yielding $\bar{\boldsymbol{x}}_{\ell+1} \approx \boldsymbol{A}\bar{\boldsymbol{x}}_\ell$ with rank $r = 10$ (App. D). Eigenvalues are concentrated just inside the unit circle and near the positive real axis, consistent with weak on-sphere contraction and predominantly angular updates; `cls` modes lie closest to $+1$ (longest memory), registers are intermediate, and patches show wider angular spread and stronger contraction (Fig. 11, right). Stacked-depth singular spectra mirror this ordering, decaying slowest for `cls` and fastest for patches. These results indicate that late depth implements low-rank, near-neutral dynamics that compress variation into a small set of collective directions while preserving long-memory channels for `cls`.

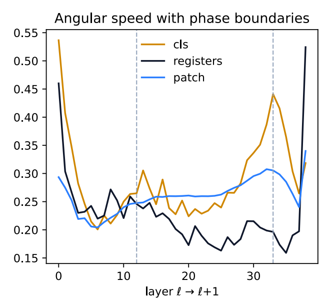

Figure 10: **Token-specific angular speed with phase overlays.** Mean angular speed s_ℓ across depth for `cls`, registers, and `patch`, with max-cut phase boundaries from Sec. 2 overlaid as vertical lines.

5 Discussion

We advanced the Block-Recurrent Hypothesis (BRH), showing empirically and constructively (via weight-tied surrogates) that recurrence can match untied baselines, and we developed *Dynamical Interpretability* by viewing depth as a flow on directions. This revealed (***i***) directional convergence to angular attractors with self-correction, (***ii***) token-specific, phase-local dynamics with punctuated transitions, and (***iii***) a late low-rank regime that coordinates updates to low dimensional subspace. While residual pathways and stochastic depth appear implicated in block recurrence, isolating causal mechanisms will require controlled training-dynamics at scale; and although two tied blocks recover most of DINOv2, a small residual gap remains that may call for improved recurrent distillation or additional time-varying components. Overall, our work highlights a recurrence induced simplicity bias, suggesting current models admit a recurrent version, implicating a potential simpler analysis. Taken together, this recurrence-induced simplicity bias and its interpretability potential point toward a broader principle: in deep learning, *recurrence finds a way*.

6 Acknowledgments

The authors would like to thank Chris Hamblin, Louis Béthune and John Hammond for many fruitful discussions. This work has been made possible in part by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute for the Study of Natural and Artificial Intelligence at

Harvard University. MJ is supported by the Kempner Institute Graduate Research Fellowship. TAK, TF, and RH are supported by the Kempner Institute Research Fellowship. DB is supported by the Kempner Institute for the Study of Natural and Artifical Intelligence at Harvard University.

References

Julius Adebayo, Michael Muelly, Ilaria Liccardi, and Been Kim. Debugging tests for model explanations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Lucian Alecu, Hugues Bonnin, Thomas Fel, Laurent Gardes, Sébastien Gerchinovitz, Ludovic Ponsolle, Franck Mamalet, Éric Jenn, Vincent Mussot, Cyril Cappi, et al. Can we reconcile safety objectives with machine learning performances? *ERTS*, 2022.

Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lucic, and Cordelia Schmid. Vivit: A video vision transformer. *ArXiv e-print*, 2021.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *Public Library of Science (PloS One)*, 2015.

Federico Baldassarre, Marc Szafraniec, Basile Terver, Vasil Khalidov, Francisco Massa, Yann LeCun, Patrick Labatut, Maximilian Seitzer, and Piotr Bojanowski. Back to the features: Dino as a foundation for video world models. *ArXiv e-print*, 2025.

Randall Balestriero et al. A spline theory of deep learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Leonard Bereska and Efstratios Gavves. Mechanistic interpretability for ai safety–a review. *ArXiv e-print*, 2024.

Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

Alexandre Boulch. Sharesnet: reducing residual network parameter number by sharing weights. *ArXiv e-print*, 2017.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017.

Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 2019.

Gregory J Chaitin. Algorithmic information theory. *IBM journal of research and development*, 1977.

Mathieu Chalvidal, Matthew Ricci, Rufin VanRullen, and Thomas Serre. Go with the flow: Adaptive control for neural odes. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Christopher Chiu, Maximilian Heil, Teresa Kim, and Anthony Miyaguchi. Fine-grained classification for poisonous fungi identification with transfer learning. *ArXiv e-print*, 2024.

Radoslaw M Cichy and Daniel Kaiser. Deep neural networks as scientific models. *Trends in cognitive sciences*, 2019.

Taco Cohen and Max Welling. Learning the irreducible representations of commutative lie groups, 2014.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *ArXiv e-print*, 2023.

Sayantan Dasgupta and Trevor Cohn. Improving language model distillation through hidden state matching. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *ArXiv e-print*, 2024.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *ArXiv e-print*, 2018.

Kamaludin Dingle, Guillermo Valle Pérez, and Ard A Louis. Generic predictions of output probability based on complexities of inputs and outputs. *Scientific reports*, 2020.

Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *ArXiv e-print*, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

Thomas Fel. *Sparks of explainability: recent advancements in explaining large vision models*. PhD thesis, Université de Toulouse, 2024.

Thomas Fel, Victor Boutin, Mazda Moayeri, Remi Cadene, Louis Bethune, Mathieu Chalvidal, and Thomas Serre. A holistic approach to unifying automatic concept extraction and concept importance estimation. *Advances in Neural Information Processing Systems (NeurIPS)*, 36, 2023.

Thomas Fel, Ekdeep Singh Lubana, Jacob S Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba Ba, and Talia Konkle. Archetypal sae: Adaptive and stable dictionary learning for concept extraction in large vision models. *Proceedings of the International Conference on Machine Learning (ICML)*, 2025.

Jesseba Fernando and Grigori Guitchounts. Transformer dynamics: A neuroscientific approach to interpretability of large language models. *ArXiv e-print*, 2025.

Ruth C. Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025.

Borjan Geshkovski, Cyril Letrouit, Yury Polyanskiy, and Philippe Rigollet. The emergence of clusters in self-attention dynamics. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

Leilani H. Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. *Proceedings of the IEEE International Conference on data science and advanced analytics (DSAA)*, 2018.

Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 1995.

Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. *ArXiv e-print*, 2023.

Liv Gorton. The missing curve detectors of inceptionv1: Applying sparse autoencoders to inceptionv1 early vision. *ArXiv e-print*, 2024.

Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *ArXiv e-print*, 2016.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 2017.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *ArXiv e-print*, 2021.

Anna Hedström, Leander Weber, Dilyara Bareeva, Franz Motzkus, Wojciech Samek, Sebastian Lapuschkin, and Marina M-C Höhne. Quantus: an explainable ai toolkit for responsible evaluation of neural network explanations. *The Journal of Machine Learning Research (JMLR)*, 2022.

Nhat M. Hoang, Do Xuan Long, Cong-Duy Nguyen, Min-Yen Kan, and Luu Anh Tuan. A comparative analysis of contextual representation flow in state-space and transformer architectures. *ArXiv e-print*, 2025.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth, 2016.

Minyoung Huh, Hossein Mobahi, Richard Zhang, Brian Cheung, Pulkit Agrawal, and Phillip Isola. The low-rank simplicity bias in deep networks. *The Journal of Transactions on Machine Learning Research (TMLR)*, 2023.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *ArXiv e-print*, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *ArXiv e-print*, 2023.

Andrei N Kolmogorov. Three approaches to the quantitative definition ofinformation'. *Problems of information transmission*, 1965.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019.

Matthew Kowal, Richard P Wildes, and Konstantinos G Derpanis. Visual concept connectome (vcc): Open world concept discovery and their interlayer connections in deep models. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Leonid Anatolevich Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 1973.

Ming Li, Paul Vitányi, et al. *An introduction to Kolmogorov complexity and its applications*. Springer, 2008.

Qianli Liao and Tomaso Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex, 2020.

Drew Linsley, Alekh Karkada Ashok, Lakshmi Narasimhan Govindarajan, Rex Liu, and Thomas Serre. Stable and expressive recurrent vision models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*. Springer, 2024.

Max Losch, Mario Fritz, and Bernt Schiele. Semantic bottlenecks: Quantifying and improving inspectability of deep representations. *International Journal of Computer Vision*, 2021.

Yiping Lu, Zhuohan Li, Di He, Zhiqing Sun, Bin Dong, Tao Qin, Liwei Wang, and Tie-Yan Liu. Understanding and improving transformer from a multi-particle dynamic system point of view. *ArXiv e-print*, 2019.

Yi Ma, Doris Tsao, and Heung-Yeung Shum. On the principles of parsimony and self-consistency for the emergence of intelligence. *Frontiers of Information Technology & Electronic Engineering*, 2022.

Chris Mingard, Henry Rees, Guillermo Valle-Pérez, and Ard A Louis. Deep neural networks have an inbuilt occam's razor. *Nature Communications*, 2025.

Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2014.

Sabine Muzellec, Leo Andeol, Thomas Fel, Rufin VanRullen, and Thomas Serre. Gradient strikes back: How filtering out high frequencies improves explanations. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2024.

Paul Novello, Thomas Fel, and David Vigouroux. Making sense of dependence: Efficient black-box explanations using dependence measure. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Chris Olah, Nick Cammarata, Chelsea Voss, Ludwig Schubert, and Gabriel Goh. Naturally occurring equivariance in neural networks. *Distill*, 2020.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *ArXiv e-print*, 2023.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *Proceedings of the International Conference on Machine Learning (ICML)*, 2013.

William Peebles and Saining Xie. Scalable diffusion models with transformers. *ArXiv e-print*, 2023.

Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *Proceedings of the British Machine Vision Conference (BMVC)*, 2018.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *ArXiv e-print*, 2021.

Michael Sander, Pierre Ablin, and Gabriel Peyré. Do residual neural networks discretize neural ordinary differential equations? *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv e-print*, 2019.

Jürgen Schmidhuber. Discovering neural nets with low kolmogorov complexity and high generalization capability. *Neural Networks*, 1997.

Jürgen Schmidhuber. The speed prior: a new simplicity measure yielding near-optimal computable predictions. *International conference on computational learning theory*, 2002.

Thiago Serra, Christian Tjandraatmadja, and Srikumar Ramalingam. Bounding and counting linear regions of deep neural networks. *Proceedings of the International Conference on Machine Learning (ICML)*, 2018.

Peter Shaw, James Cohan, Jacob Eisenstein, and Kristina Toutanova. Bridging kolmogorov complexity and deep learning: Asymptotically optimal description length objectives for transformers. *ArXiv e-print*, 2025.

Sam Shleifer and Alexander M Rush. Pre-trained summarization distillation. *ArXiv e-print*, 2020.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

Shawn Tan, Yikang Shen, Zhenfang Chen, Aaron Courville, and Chuang Gan. Sparse universal transformer. *ArXiv e-print*, 2023.

Matus Telgarsky. Representation benefits of deep feedforward networks. *ArXiv e-print*, 2015.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024.

Vincent Tjeng and Russ Tedrake. Verifying neural networks with mixed integer programming. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Andreas Veit, Michael J Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, et al. Recursive self-aggregation unlocks deep thinking in large language models. *ArXiv e-print*, 2025.

Martina G Vilas, Timothy Schaumlöffel, and Gemma Roig. Analyzing vision transformers for image classification in class embedding space. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yue Wang and Sai Ho Chung. Artificial intelligence in safety-critical systems: a systematic review. *Industrial Management & Data Systems*, 2022.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1989.

Andrew Gordon Wilson. Deep learning is not so mysterious or different. *ArXiv e-print*, 2025.

Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S. Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks, 2019.

Qing Yun. Vision transformers (vits) for feature extraction and classification of ai-generated visual designs. *IEEE Access*, 2025.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*, 2014.

A Appendix

B Training Block Recurrent Foundation Models

Model Architecture. All Raptor variants ($k = 2, 3, 4$) are trained on top of DINOv2 (ViT-B with registers) (Darcet et al., 2023) and use the same transformer architecture:

- Feature dimension: 784
- MLP ratio: 4
- Multi-head attention heads: 12

The depth-scale MLP consists of a linear layer expanding from dimension 1 to 16, followed by a SiLU activation, and a second linear layer mapping from 16 to 3 × dim. This produces three separate scaling vectors.

Below, we provide the hyperparameters and training settings used to train Raptor. We first describe the layer divisions for different values of k, followed by the training procedure.

Layer Splits. For each choice of k, the encoder layers are divided into blocks as follows:

k	Block	Input → Predicted Layers
2	Block 1	0 → 1–7
	Block 2	7 → 8–12
3	Block 1	0 → 1–7
	Block 2	7 → 8–10
	Block 3	10 → 11–12
4	Block 1	0 → 1–4
	Block 2	4 → 5–7
	Block 3	7 → 8–10
	Block 4	10 → 11–12

Table 3: Layer splits used for training with different values of k.

We select these partitions using the max cut algorithm applied to the DINOv2 ViT-B activation layer-layer cosine similarity matrix using the ImageNet-1k validation set.

Stage 1: Independent Block Training. Each block is trained independently on the subset of layers it is responsible for predicting. For example, when $k = 3$, Block 1 is trained to predict Layers 1–7. Training details are summarized in Table 4.

Stage 2: Joint Training. After independent training, all blocks are connected to autoregressively predict Layers 1–12 end-to-end. Each block still predicts its designated segment, but the entire model now backpropagates through the full sequence. Training details are summarized in Table 5.

Loss Function with weight on `cls` token For a given set of ground truth DINOv2 activations $x \in \mathbb{R}^{N \times D}$, where N is the number of tokens and D is the embedding dimension, we use the following to calculate the mean squared error between predictions by Raptor $\hat{x}$ and x:

$$\mathcal{L} = \lambda_{CLS} \left\| \hat{x}_{CLS} - x_{CLS} \right\|^2 + \lambda_{REG} \left\| \hat{x}_{REG} - x_{REG} \right\|^2 + \lambda_{PATCH} \left\|^2 \hat{x}_{PATCH} - x_{PATCH} \right\|^2, \quad (4)$$

Setting	Value
Dataset	ImageNet-1k (train split)
Epochs	20
Batch Size	64
Optimizer	AdamW
Weight Decay	0.0001
Learning Rate Schedule	Linear warmup (10,000 steps) to 1×10^{-4}, then cosine decay to 1×10^{-6}
Teacher Forcing Loss Weight (annealed, first 5 epochs)	$\lambda : 0.5 \to 0$
Block 3 Token Loss Weights	$\lambda_{CLS} = 0.34$, $\lambda_{REG} = 0.33$, $\lambda_{PATCH} = 0.33$

Table 4: Stage 1 training hyperparameters for block-wise training.

Setting	Value
Dataset	ImageNet-1k (train split)
Epochs	20
Batch Size	64
Weight Decay	0.0001
Optimizer	AdamW
Learning Rate Schedule	Same as Stage 1
Token Loss Weights	$\lambda_{CLS} = 0.45$, $\lambda_{REG} = 0.1$, $\lambda_{PATCH} = 0.45$

Table 5: Stage 2 joint training hyperparameters.

B.1 Phase discovery via a contiguous max-cut on the layer–layer similarity

Problem setup. Let $S \in \mathbb{R}^{L \times L}$ be the (symmetrized) layer-layer similarity matrix, where S_{ij} measures the similarity between layers i and j (for example, cosine similarity). We seek a partition of depth into k contiguous segments or "phases". $\Pi = \{[b_1, e_1], \ldots, [b_k, e_k]\}$ with $1 = b_1 \leq e_1 < b_2 \leq e_2 < \cdots < b_k \leq e_k = L$ and $e_t + 1 = b_{t+1}$, that maximizes within-block similarity (equivalently, minimizes cross-block cut).

Objectives. For a segment $[i, j]$ of length $n = j - i + 1$, define:

$$\mathrm{sum}(i,j) = \sum_{p=i}^{j} \sum_{q=i}^{j} S_{pq}, \quad \mathrm{offdiag}(i,j) = \mathrm{sum}(i,j) - \sum_{p=i}^{j} S_{pp}.$$

We consider additive segment scores $g(i, j)$ by computing the final weighted mean as:

$$\frac{\mathrm{sum}(i,j)}{n^2};$$

Maximizing $\sum_{t=1}^{k} g(b_t, e_t)$ prefers blocks that are internally similar and, by contiguity, implies small cross-block interfaces (a contiguous max-cut on the line).

Fast block queries via 2-D prefix sums. Precompute a 2-D prefix (summed-area) table $P \in \mathbb{R}^{(L+1)\times(L+1)}$ with $P_{rc} = \sum_{u<r} \sum_{v<c} S_{uv}$. Then any submatrix sum obeys

$$\mathrm{sum}(i,j) = P_{j+1,j+1} - P_{i,j+1} - P_{j+1,i} + P_{i,i},$$

in $O(1)$ time; diagonal sums use a 1-D prefix over $\mathrm{diag}(S)$. This is sometimes referred to as the "integral image" trick.

Contiguous DP solver ($O(kL^2)$). Let $\mathrm{dp}[t, j]$ be the best score for partitioning layers $1..j$ into t blocks. With minimum block length m,

$$\mathrm{dp}[1,j] = g(1,j) \quad (j \geq m), \qquad \mathrm{dp}[t,j] = \max_{i \in \{t\,m-1, \ldots, j-m\}} \mathrm{dp}[t-1,i] + g(i+1,j),$$

for $t = 2, \ldots, k$ and $j \geq t\,m$. We keep backpointers $\mathrm{prev}[t, j]$ to recover boundaries by backtracking from $(t{=}k, j{=}L)$. With $g(\cdot)$ evaluated in $O(1)$ by prefix sums, the overall complexity is $O(kL^2)$ time and $O(kL)$ memory. This DP structure mirrors classical optimal 1-D segmentation/partitioning solvers.

B.2 Teacher-student reconstruction R^2

To stabilize measures of pairwise vector similarity over large hyperparameter sweeps when fits may be poor, we use an alternative calculation for R^2 for small-scale models (Figure 4). Here, we first regress the student's `cls` or `patch` tokens $\in R^{N\times D}$ to the corresponding teacher tokens $\in R^{N\times D}$ using ordinary least squares with a bias term. N is the number of tokens and D is the dimensionality of the token. We then calculate the average of the R^2 values between the true teacher token vectors and the student's reconstruction of those vectors. Note that this regression is purely a 1-dimensional rescaling and shifting for each student token vector. This results in an R^2 value that is bounded between 0 and 1, and can be understood to present the 'explainable variance' between the student and teacher representations.

B.3 Linear Probe Fine-tuning

We fine-tune linear probes on three downstream datasets: ImageNet-1k (classification), ADE20k (semantic segmentation), and NYUv2 (monocular depth estimation).

For ImageNet-1k and ADE20k, we use the AdamW optimizer with linear warmup followed by cosine learning rate decay. For NYUv2, we use AdamW with `GradScaler` and mixed precision training. All probes operate on the final block's prediction of Layer 12, using either the `cls` token, patch tokens, or both, depending on the task. The detailed hyperparameters are shown in Table 6.

For NYUv2, we adopt an approach similar to Oquab et al. (2023). Specifically, we use images at a 480×640 resolution and center pad them so that the dimensions are multiples of 14. We feed the images through the model and extract the predictions from the final layer. The `cls` token is concatenated with the patch tokens, and the spatial resolution is upsampled by a factor of 4. Both the `cls` and patch tokens are upscaled, after which the `cls` token is concatenated to each patch token. We treat this representation as the "logits." To obtain depth, we normalize the logits with a softmax and compute the weighted average of the centers of 256 evenly spaced bins. Then, we upsample this representation to 480×640 and consider the result our depth. For training, we use the loss function introduced by Bhat et al. (2021).

Hyperparameter	ImageNet-1k	ADE20k	NYUv2
Epochs	15	10	25
Batch Size	512	32	128
Base LR	1×10^{-3}	1×10^{-3}	1×10^{-4}
Weight Decay	1×10^{-2}	1×10^{-2}	1×10^{-2}
Grad. Clip Norm	1.0	1.0	1.0
Warmup Iters	100	100	100
Optimizer	AdamW	AdamW	AdamW + GradScaler
Head Init.	DINOv2 classification probe	Random segmentation head	Random depth head
Input Tokens	concat(`cls`, mean patch)	Patch	concat(`cls`, patch)

Table 6: Linear probe fine-tuning hyperparameters across datasets. Base LR denotes the peak learning rate before cosine decay.

C Dynamics Protocols and Metrics

This appendix consolidates definitions and experimental procedures used in Sec. 4. All measurements are performed on ImageNet validation data. For aggregate statistics, we use 10k randomly sampled validation images. For trajectory visualizations (e.g., Fig. 7), we select five ImageNet classes with $1{,}000$ images each. Inputs are resized to 256 pixels on the shorter side and center-cropped to 224×224. Unless otherwise noted, we use `DINOv2-Giant` with four register tokens from the official implementation.

Normalization. Token states $\boldsymbol{x}_\ell \in \mathbb{R}^d$ at depth ℓ are decomposed into norm and direction. We study normalized states

$$\hat{\boldsymbol{x}}_\ell = \frac{\boldsymbol{x}_\ell}{\|\boldsymbol{x}_\ell\|} \in \mathbb{S}^{d-1},$$

so that dynamics are restricted to the unit sphere.

Directional convergence. Directional similarity to the terminal representation is measured by

$$\gamma_\ell = \langle \hat{\boldsymbol{x}}_\ell, \hat{\boldsymbol{x}}_L \rangle,$$

which traces the angular alignment of layer ℓ to the final state.

Angular speed. Per-layer angular update magnitude is defined as

$$s_\ell = \arccos\langle \hat{\boldsymbol{x}}_{\ell+1}, \hat{\boldsymbol{x}}_\ell \rangle.$$

Statistics of s_ℓ are stratified by token type.

Phase overlays. Phase boundaries are obtained from the max-cut segmentation of representational similarity matrices (Sec. 2) and used as vertical markers in angular speed and sensitivity plots.

Perturbation protocol. To probe stability, we add a perturbation $\varepsilon \boldsymbol{u}$ at layer ℓ,

$$\tilde{\boldsymbol{x}}_\ell = \boldsymbol{x}_\ell + \varepsilon \boldsymbol{u}, \quad \boldsymbol{u} \sim \mathcal{N}(0, I_d),$$

and follow the normalized trajectory thereafter. Sensitivity is quantified by the terminal cosine deviation

$$d_{\cos}(\hat{\boldsymbol{x}}_L^{(\varepsilon,\ell)}, \hat{\boldsymbol{x}}_L) = 1 - \langle \hat{\boldsymbol{x}}_L^{(\varepsilon,\ell)}, \hat{\boldsymbol{x}}_L \rangle.$$

Low-rank and coherence metrics. For angular updates $\Delta_\ell^{(i)} = \hat{\boldsymbol{x}}_{\ell+1}^{(i)} - \hat{\boldsymbol{x}}_\ell^{(i)}$, we form the update matrix $\boldsymbol{U}_\ell$. Stable rank is given by

$$r_s(\boldsymbol{U}_\ell) = \frac{\|\boldsymbol{U}_\ell\|_F^2}{\|\boldsymbol{U}_\ell\|_2^2},$$

and coherence by

$$\kappa_\ell = \tfrac{1}{N} \sum_i \frac{\langle \Delta_\ell^{(i)}, \bar{\Delta}_\ell \rangle}{\|\Delta_\ell^{(i)}\| \, \|\bar{\Delta}_\ell\|}, \quad \bar{\Delta}_\ell = \tfrac{1}{N} \sum_i \Delta_\ell^{(i)}.$$

D Dynamic Mode Decomposition

Let $\boldsymbol{f}$ be a trained ViT with transformer layers $\{\boldsymbol{f}_\ell\}_{\ell=1}^L$. For $x \in \mathcal{X}$, denote by $\boldsymbol{A}_\ell(x) \in \mathbb{R}^{T \times d}$ the token matrix at depth ℓ with $T = 1 + R + P$ (`cls`, R registers, P `patch`). Form group states by within-layer averaging

$$\boldsymbol{z}_\ell^{(\mathsf{cls})}(x) = \boldsymbol{A}_\ell(x)_{\mathsf{cls}} \quad \boldsymbol{z}_\ell^{(\mathsf{reg})}(x) = \tfrac{1}{R} \sum_{t \in \mathcal{T}_{\mathsf{reg}}} \boldsymbol{A}_\ell(x)_t \quad \boldsymbol{z}_\ell^{(\mathsf{patch})}(x) = \tfrac{1}{P} \sum_{t \in \mathcal{T}_{\mathsf{patch}}} \boldsymbol{A}_\ell(x)_t$$

and enforce per-layer ℓ_2 normalization on the group averages

$$\boldsymbol{x}_\ell^{(g)}(x) = \frac{\boldsymbol{z}_\ell^{(g)}(x)}{\|\boldsymbol{z}_\ell^{(g)}(x)\|_2} \in \mathbb{S}^{d-1} \subset \mathbb{R}^d.$$

All DMD fits below are performed independently for each $g \in \{\mathtt{cls}, \mathtt{reg}, \mathtt{patch}\}$ on the depth trajectory $\boldsymbol{x}_{0:L}^{(g)}(x)$. We start by stacking states along depth to form

$$\boldsymbol{Y}^{(g)} = \begin{bmatrix} (\boldsymbol{x}_0^{(g)})^\top \\ \vdots \\ (\boldsymbol{x}_L^{(g)})^\top \end{bmatrix} \in \mathbb{R}^{(L+1) \times d} \quad \boldsymbol{X}_1 = (\boldsymbol{Y}_{0:L}^{(g)})^\top \in \mathbb{R}^{d \times L} \quad \boldsymbol{X}_2 = (\boldsymbol{Y}_{1:L+1}^{(g)})^\top \in \mathbb{R}^{d \times L}.$$

DMD fits a single linear depth-step map $\boldsymbol{A}$ with $\boldsymbol{X}_2 \approx \boldsymbol{A}\boldsymbol{X}_1$.

Exact DMD at rank r. Compute the thin SVD $\boldsymbol{X}_1 = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ and select $r \le \operatorname{rank}(\boldsymbol{X}_1)$. Let $\boldsymbol{U}_r$ $\boldsymbol{\Sigma}_r$ $\boldsymbol{V}_r$ be the leading blocks and define the reduced operator

$$\widetilde{\boldsymbol{A}} = \boldsymbol{U}_r^\top \boldsymbol{X}_2 \boldsymbol{V}_r \boldsymbol{\Sigma}_r^{-1} \in \mathbb{R}^{r\times r}.$$

Diagonalize $\widetilde{\boldsymbol{A}}\boldsymbol{W} = \boldsymbol{W}\boldsymbol{\Lambda}$ with $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \dots, \lambda_r) \in \mathbb{C}^{r\times r}$ and $\boldsymbol{W} \in \mathbb{C}^{r\times r}$. The exact DMD modes in ambient space are

$$\boldsymbol{\Phi} = \boldsymbol{X}_2 \boldsymbol{V}_r \boldsymbol{\Sigma}_r^{-1} \boldsymbol{W} \in \mathbb{C}^{d\times r}.$$

Modal amplitudes for the initial state are $\boldsymbol{b} = \boldsymbol{\Phi}^\dagger \boldsymbol{x}_0^{(g)} \in \mathbb{C}^r$. One-step and t-step reconstructions are

$$\widehat{\boldsymbol{x}}_1^{(g)} \approx \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{b} \quad \widehat{\boldsymbol{x}}_t^{(g)} \approx \boldsymbol{\Phi}\boldsymbol{\Lambda}^t\boldsymbol{b} \text{ for } t \ge 0$$

with the induced linear predictor $\boldsymbol{A} = \boldsymbol{X}_2 \boldsymbol{V}_r \boldsymbol{\Sigma}_r^{-1} \boldsymbol{U}_r^\top$. No affine offset is fitted.

On-sphere interpretation. Because each $\boldsymbol{x}_\ell^{(g)}$ lies on $\mathbb{S}^{d-1}$ the map $\boldsymbol{A}$ is a best linear approximation of the depth flow restricted to the unit sphere. For $\lambda_i = |\lambda_i| e^{\mathrm{i}\theta_i}$ the modulus $|\lambda_i|$ measures contraction of directions within the span of modes on the sphere and the angle θ_i captures per-layer rotational change. The spectral radius $\rho(\boldsymbol{A}) = \max_i |\lambda_i|$ and the median of $|\lambda_i|$ summarize contraction strength. In particular this explain why all the eigenvalue in Figure 11 are contain in $\mathbb{S}^2$, see the report of the eigenvalue cloud $\{\lambda_i\}_{i=1}^r$ in the complex plane and the singular spectrum $\{\sigma_i\}_{i=1}^r$ of $\boldsymbol{X}_1$.

Appendix: Levin Complexity under 0-BRH

Background & Positioning. There is a long line of work linking plain Kolmogorov complexity Kolmogorov (1965); Chaitin (1977) to its resource-bounded variants, notably Levin's time-bounded measure, which penalizes programs by both description length and runtime (the $|p| + \log T$ form) (Levin, 1973; Li et al., 2008). Deep learning research has not remained isolated from these ideas: connections have been explored both for discovering simpler neural networks Schmidhuber (1997) and for contextualizing runtime priors Schmidhuber (2002) within neural architectures.

In our case, the proof is a direct application of this standard framework as formalized in Li et al. (2008): once we exhibit a prefix-free program p that computes $\boldsymbol{f}_\ell$ and bound its running time $T(p)$, Levin complexity immediately yields $K_{\text{Levin}}(\boldsymbol{f}_\ell) \le |p| + \log T(p) + O(1)$. The only domain-specific ingredients are *(i)* a BRH-based encoding that prices the tied blocks and schedule at $\sum_{j=1}^k DL_U(\boldsymbol{\theta}(\boldsymbol{B}_j)) + O(k\log\ell)$ using standard self-delimiting integer codes, and *(ii)* a runtime-parity assumption linking $T(p)$ to the model's untied runtime $R(\boldsymbol{f}_\ell)$ up to machine and invariance constants. Formally:

Proposition 2 (BRH guarantees low Levin complexity)**.** *Let $\boldsymbol{f}_\ell$ satisfy 0-BRH with $k \ll \ell$ tied blocks $\{\boldsymbol{B}_j\}_{j=1}^k$ and schedule $(n_j)_{j=1}^k$ with $\sum_j n_j = \ell$. Let $R(\cdot)$ denote block runtime and define the untied teacher runtime $R(\boldsymbol{f}_\ell) := \sum_{j=1}^k n_j R(\boldsymbol{B}_{\boldsymbol{f}})$; assume runtime parity $R(\boldsymbol{B}_j) \le (1+\delta) R(\boldsymbol{B}_{\boldsymbol{f}})$. Then*

$$K_{\text{Levin}}(\boldsymbol{f}_\ell) \le \sum_{j=1}^k DL(\boldsymbol{\theta}(\boldsymbol{B}_j)) + O(k\log\ell) + \log R(\boldsymbol{f}_\ell) + O(1).$$

Proof. We follow the standard time-bounded description framework of Levin (1973), and contextualized it to the 0-BRH setting (perfect reconstruction along the entire depth trajectory), in the spirit of the sketch already outlined in the paper. We fix a universal prefix-free machine U and adopt the same finite-precision arithmetic used by the target model to define $DLU(\cdot)$; this ensures that encoding parameters with length $DLU(\boldsymbol{\theta}(\boldsymbol{B}_j))$ bits suffices to reproduce the exact numerical behavior of the corresponding block under that arithmetic. With this convention, "exactly computes" means bit-for-bit equivalence with the reference implementation of $\boldsymbol{f}_\ell$ (under its inference precision).

The proof proceeds in three steps: we first bound the description length, then verify correctness (equality of the program output), and finally bound the runtime before combining these results to derive the final bound.

The first step consists in bounding the description length. We describe a prefix-free program p that parses (i) the tied block parameters and (ii) a self-delimiting schedule, with total length

$$|p| \;\le\; \sum_{j=1}^{k} DLU(\boldsymbol{\theta}(\boldsymbol{B}_j)) \;+\; O(k\log\ell) \;+\; O(1).$$

For each tied block, encode its parameters using the scheme underlying DLU, yielding a prefix-free code of length $DLU(\boldsymbol{\theta}(\boldsymbol{B}_j))$. Concatenate the k codes and prepend an $O(1)$ header that instructs U how to parse and reconstruct each $\boldsymbol{B}_j$ from its bitstream. Encode the schedule $(n_1,\ldots,n_k)$ and the value of k as self-delimiting integers (any standard universal code suffices), which costs $O(k\log\ell)$ bits because $n_j \le \ell$ and $\sum_j n_j = \ell$. The parser is a constant-size routine bundled with p, absorbed into $O(1)$.

Now we show that this construction is a valid under 0-BRH. Let $g_\ell := \boldsymbol{B}_k^{(n_k)} \circ \cdots \circ \boldsymbol{B}_1^{(n_1)}$ denote the composed tied map realized by iterating the reconstructed blocks according to the parsed schedule. By 0-BRH, we have exact functional equality

$$\boldsymbol{f}_\ell \;=\; \boldsymbol{B}_k^{(n_k)} \circ \cdots \circ \boldsymbol{B}_1^{(n_1)} \;=\; g_\ell,$$

i.e., the tied computation reproduces the full internal trajectory (not only the terminal output). Because p encodes the same block parameters at the precision that defines DLU and U evaluates precisely the same composition, we obtain, for all inputs x,

$$U(p)(x) = g_\ell(x) = \boldsymbol{f}_\ell(x).$$

Finally, we get the runtime bound to derive the Levin Complexity. On the fixed machine U, a single evaluation of block $\boldsymbol{B}_j$ incurs time at most $c\,R(\boldsymbol{B}_j)$ for a machine dependent constant $c > 0$ (standard machine invariance up to constants; cf. the invariance discussion in Kolmogorov/Levin-style arguments (Kolmogorov, 1965; Levin, 1973)). Executing the schedule therefore costs

$$T(p) \;\le\; c\sum_{j=1}^{k} n_j\,R(\boldsymbol{B}_j).$$

Applying the runtime parity assumption $R(\boldsymbol{B}_j) \le (1+\delta)\,R(\boldsymbol{B}_{\boldsymbol{f}})$ for all j and using $R(\boldsymbol{f}_\ell) := \sum_{j=1}^{k} n_j R(\boldsymbol{B}_{\boldsymbol{f}})$ gives

$$T(p) \;\le\; c\,(1+\delta)\sum_{j=1}^{k} n_j\,R(\boldsymbol{B}_{\boldsymbol{f}}) \;=\; c\,(1+\delta)\,R(\boldsymbol{f}_\ell).$$

By the definition of Levin's time-bounded complexity

$$K_{\text{Levin}}(\boldsymbol{f}_\ell) \;\le\; |p| + \log T(p) + O(1),$$

and combining with the bounds from the previous steps yields

$$K_{\text{Levin}}(\boldsymbol{f}_\ell) \;\le\; \sum_{j=1}^{k} DLU(\boldsymbol{\theta}(\boldsymbol{B}_j)) \;+\; O(k\log\ell) \;+\; \log R(\boldsymbol{f}_\ell) \;+\; \underbrace{\log c + \log(1+\delta) + O(1)}_{=O(1)}.$$

Absorbing the machine constant and the parity slack into $O(1)$ gives the stated inequality. □

E Supplementary Results

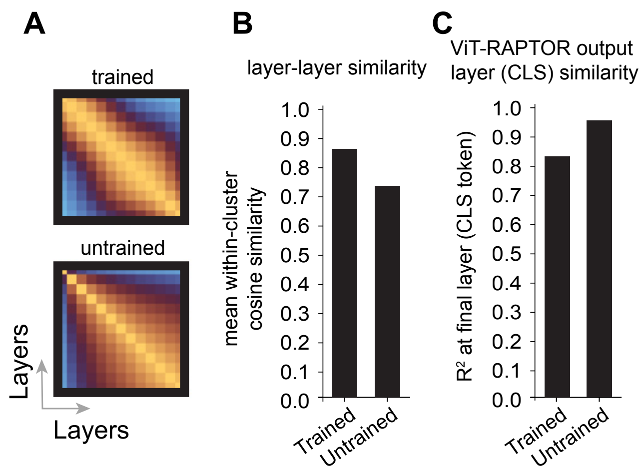

Figure 12: **Training increases layer-layer similarity and teacher-student reconstruction. A)** Layer-layer cosine similarity matrices of a trained ViT (top) and untrained ViT (bottom). Network trained with 0.3 uniform probability stochastic depth. **B)** Mean intra-block cosine similarity with max-cut k=3 for trained and untrained ViTs. **C)** Teacher-student `Raptor` reconstruction of output `cls` token for trained and untrained ViTs.