Two Speeds of Learning: A Representation-Readout Decomposition of Grokking and Double Descent

Chi-Ning Chou* Harvard University & Flatiron Institute

Oscar Uzdelewicz* New York University

Neng-Chun Chiu Harvard University

Yao-Yuan Yang Thinking Machines Lab

SueYeon Chung Harvard University & Flatiron Institute

Abstract

Training loss and accuracy are the standard signals used to monitor generalization during deep neural network training. Two well-documented phenomena complicate this picture: in grokking, train loss falls rapidly while test performance improves abruptly only after a long delay; in epoch-wise double descent, train loss decreases monotonically while test loss or error rises and falls. Existing accounts are often task-specific, and a task-agnostic analysis framework for diagnosing and explaining these phenomena across realistic tasks and architectures is missing. We address this challenge by analyzing two competing processes that underlie learning dynamics: representation learning in the encoder and readout calibration in the final classifier. Using tools from representational geometry, neural tangent kernels, and linear probing, we show that both processes are active throughout training, with the fluctuations of their relative speed giving rise to seemingly anomalous generalization dynamics. Applying the representation-readout decomposition to grokking across a wide range of tasks and architectures, we find that the readout is train-biased before grokking onset, and representation learning is gradual but not absent—contrary to the lazy-to-rich account. The framework further provides diagnostic signatures distinguishing spurious from genuine generalization: in a previously reported MNIST grokking example and an epoch-wise double descent example, apparent delayed or non-monotone generalization is shown to arise from representation degradation and readout misalignment induced by non-standard training recipes. Together, these results establish the representation-readout decomposition as a top-down framework for understanding learning dynamics and revealing underlying algorithms for interpretability research.

1 Introduction

Training loss and accuracy are the standard signals used to monitor progress during deep neural network training [1]. In practice, we monitor these signals to assess whether a model is properly learning, decide when to stop training, and forecast eventual test performance. Two well-documented phenomena complicate this picture. In *grokking* [2, 3, 4, 5], a model first achieves near-perfect performance on the training objective while test accuracy remains near chance level, and only much later does generalization appear to emerge abruptly (Figure 1b, top). In *epoch-wise double descent* [6, 7], train loss decreases monotonically while test loss rises and falls non-monotonically over the course of training (Figure 1b, bottom).

*These authors contributed equally as first authors. Contact: {chiningchou, schung}@fas.harvard.edu.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

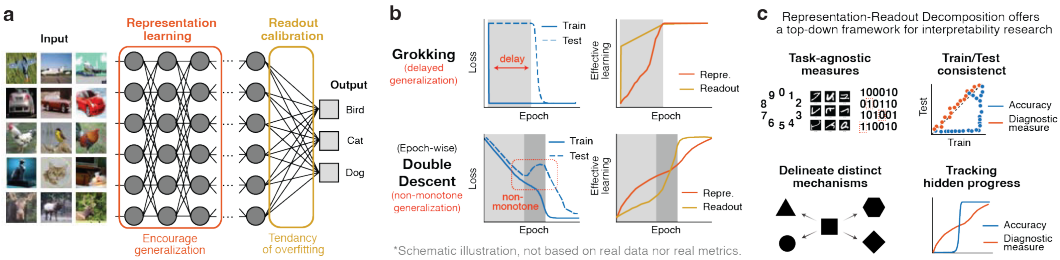

Figure 1: **Overview of our framework. a**. A neural network can be conceptually decomposed into two parts: a *feature learning* part (orange) that encourages generalization, and a *readout calibration* part (yellow) that is prone to overfitting (see Section 2 for details). **b**. We propose that grokking (top) and epoch-wise double descent (bottom) can be explained by the decoupling between the dynamics of representation learning and readout calibration. **c**. Our proposed representation-readout decomposition offers an analysis framework for future interpretability research.

These phenomena raise two linked concerns. Practically, they undermine the standard signals used to monitor training and forecast generalization. Scientifically, they have been interpreted as evidence of phase transitions or other qualitative shifts in learning dynamics. Much existing work addresses these phenomena case by case, with task-specific progress measures for particular grokking settings [8, 9] and tailored mechanistic accounts for individual phenomena [3, 10]. Addressing both concerns calls for a unified, task-agnostic framework with reliable observables for monitoring generalization across diverse tasks and architectures.

An exciting recent task-agnostic account for grokking is the kernel-based *lazy-to-rich* framework [10], which attributes grokking to a transition from the lazy regime—where features are essentially fixed, and readout does the heavy lifting—to the rich regime—where features actively reorganize for better generalization. In parallel, another recent line of work [11, 12, 13] uses representation geometry to reveal properties of learning dynamics invisible to kernel-based measures. Building on these two perspectives, we propose to track the learning dynamics through an explicit decomposition of the network into an *encoder* ϕ_θ that maps inputs to representations and a *readout matrix* W that maps representations to logits (Figure 1a). The representation-readout decomposition enables new insights into two speeds[2] of learning in grokking and double descent:(Figure 1b)—regarding their underlying principles (Section 3) and spurious modes (Section 4)—and establishes a top-down[3] framework for studying learning dynamics that can enrich future mechanistic interpretability research (Figure 1c).

1.1 Our contributions

A representation-readout decomposition for learning dynamics. We propose to study the learning dynamics of deep neural networks through an explicit decomposition into an encoder and a readout, tracking the two separately throughout training (Section 2). The diagnostic measures we use—critical dimension, task-relevant geometric measures [14, 11, 15], linear probing [16], and Neural Tangent Kernel [17]—are existing tools, but their joint use within the decomposition yields a new vocabulary for describing learning dynamics, going beyond the binary lazy/rich characterization of [10].

Two speeds of learning in grokking and double descent. Applied to canonical grokking examples (e.g., modular addition [2, 9, 18], permutation composition [19, 20], and sparse parity [8, 21]), our analysis refines the previous lazy-to-rich account: representation learning is not silent in early epochs, it is just slower than the speed[2] of readout calibration (Section 3). Moreover, the underlying improvement in representation is gradual and persistent rather than sudden. Moreover, the diagnostic measures discriminate genuine grokking phenomena from spurious cases: a previously reported grokking example on MNIST [3] is revealed as a readout artifact, in which the apparent delayed generalization arises because of representation quality degrading in early epochs under ill-posed training setups (Section 4.1). The same analysis sheds light on epoch-wise double descent (Section 4.2).

[2]Speed here refers to relative rate of progress over training, inferred from the diagnostic measures in Section 2.2.

[3]Here, top-down refers to characterizing the global structure of representations, e.g., through manifold geometry, separability, and alignment, as opposed to bottom-up approaches such as reverse-engineering circuits or features.

A top-down analysis framework for interpretability research. The decomposition and its diagnostic measures offer a top-down[3] analysis framework that advances existing bottom-up approaches to mechanistic interpretability research. Unlike circuit-level analyses, our framework is task-agnostic: the diagnostic measures are consistent across train and test splits, track generalization reliably throughout training, and the geometric vocabulary they provide can describe and discriminate qualitatively different mechanisms at a structural level (Figure 1c and Figure 5). We anticipate that this framework will be useful for identifying generalization failures, diagnosing unsafe or spurious learning dynamics, and guiding mechanistic investigations in more complex and safety-critical settings.

1.2 Related work

Mechanistic explanations of grokking and double descent. The phenomenon of grokking was first reported by Power et al. [2], who observed delayed generalization in modular arithmetic tasks despite rapid convergence of training loss. Subsequent work has demonstrated grokking-like dynamics across a range of synthetic and real-world settings by varying weight decay value and/or scaling up the loss value [3]. A substantial body of research has sought mechanistic explanations for this behavior. Analytical studies of simplified models, including sparse parity and Fourier-based analyses [8, 9], characterize regimes in which delayed generalization arises from the interaction between optimization and representational structure. Empirical investigations have connected grokking to feature learning dynamics [10], circuit efficiency and competition [22, 21], representational geometry [13], and evolving complexity or robustness properties [4, 23]. Double descent was introduced by Belkin et al., [6] in the model-wise setting and characterized in the epoch-wise setting by Nakkiran et al., [7] through introducing label noises, and subsequent analyses have attributed it to slower learning dynamics [24]. Together, these works illuminate when and how grokking and double descent can occur by analyzing changes in the underlying learning dynamics and representations. See Appendix C.1,C.2,C.4 for more discussion.

Progress measures for grokking. Several studies have proposed specific progress measures to track generalization in grokking regimes, including Fourier-based [8, 9], weight-based [3], kernel-based [10], and representation-based [25, 13] progress measures. In contrast, the measures used in our analysis framework go beyond tracking progress toward generalization: they diagnose the state of the learning dynamics and reveal distinct underlying forces such as representation degradation and readout misalignment. This is why we refer to them as *diagnostic* rather than *progress* measures. See Section C.3 for more discussion.

Lazy-to-rich account for grokking. The lazy and rich training regimes are grounded in the Neural Tangent Kernel (NTK) theory [26, 27]: in the lazy regime, the network behaves as a kernel method with fixed initial features; in the rich regime, features actively reorganize beyond the linearization. [10] propose that grokking arises from a transition between these two regimes, identifying two key determinants: a laziness parameter that controls the rate of feature learning by scaling the network output, and the alignment between the initial NTK and the task labels. They develop this theory in the setting of polynomial regression with a two-layer MLP and vanilla gradient descent. The scale factor view also mathematically coincides with the other approach to induce grokking via large initial weight norm [3]. Our work builds directly on this account but refines it: rather than an abrupt binary transition, we show that representation learning is slow but continuously active throughout training (Section 3.1), and that the readout remains systematically train-biased until representations are rich enough to support generalization (Section 3.2). See Section C.1 for more discussion.

2 Methods

In this section, we formally set up the representation-readout decomposition (Section 2.1), the task-agnostic measures used in the analysis (Section 2.2), and the experiments (Section 2.3).

2.1 Representation-Readout Decomposition

We consider a neural network as a function $f : \mathcal{X} \to \mathbb{R}^C$, where $\mathcal{X}$ is the input domain (e.g., the pixel space of natural images, or the group $\mathbb{Z}/n\mathbb{Z}$ for modular arithmetic) and C is the dimension of the output space (e.g., the number of classes/labels for classification). Outputs are interpreted as

logits, with prediction $\arg\max_i f_i(\mathbf{x})$. See Section 2.3 for the specific tasks and architectures studied in this paper, and see Appendix B for full details.

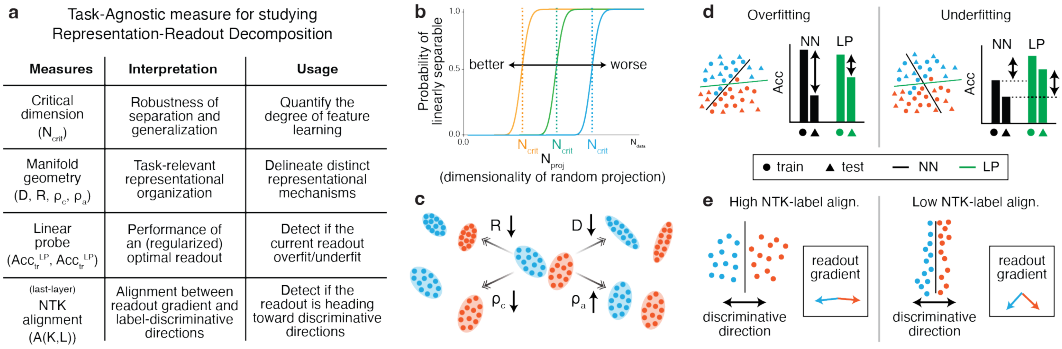

Figure 2: **Task-agnostic diagnostic measures for the representation-readout decomposition. a**, Summary of the four measures, their interpretations, and their diagnostic usage. **b**, The critical dimension N_{crit} is defined as the median threshold of the random-projection separability probability $p(N_{\text{proj}})$; a smaller N_{crit} indicates more robustly untangled manifolds. **c**, Four task-relevant geometric measures characterize the underlying geometric mechanisms for separating/untangling label-conditioned representation manifolds $\{\mathcal{M}_c\}$. **d**, The linear-probe gap compares the best regularized linear readout to the actual readout. **e**, Last-layer NTK alignment measures whether the readout's gradient updates point along label-discriminative directions.

We decompose f as the composition of a nonlinear *encoder* ϕ_θ and a linear *readout* W:

$$f(\mathbf{x}) = W\,\phi_\theta(\mathbf{x}), \qquad \phi_\theta : \mathcal{X} \to \mathbb{R}^N, \quad W \in \mathbb{R}^{C\times N}. \tag{1}$$

Here ϕ_θ folds all upstream parameters θ into a single map to an N-dimensional representation, and W is the final linear layer mapping representations to logits.[4]

Our choice to decompose the network at the final linear layer is deliberate, motivated by the elegance and generality it affords the analysis. We adopt it because the linear readout is a canonical architectural element shared across MLPs, CNNs, and transformers, and because it allows us to define task-agnostic measures to probe learning dynamics (Section 2.2). While one could in principle decompose at intermediate layers, we defer a systematic exploration of such choices to future work. See also Figure 1a for a schematic illustration. Below are examples of representation-readout decompositions for MLP and Transformer architectures.

MLP example. For an L-layer MLP with the penultimate layer width N and element-wise nonlinearity σ, set $\mathbf{h}_0 = \mathbf{x}$ and $\mathbf{h}_\ell = \sigma(W_\ell \mathbf{h}_{\ell-1})$, where $\ell = 1, \ldots, L-1$. The decomposition is then $\phi_\theta(\mathbf{x}) = \mathbf{h}_{L-1}$ and $W = W_L$, with $\theta = \{W_\ell\}_{\ell=1}^{L-1}$. The encoder is the network up to the penultimate layer's activations; the readout is the final linear layer.

Transformer example. For a transformer with L attention-plus-MLP blocks, the decomposition takes $\phi_\theta(\mathbf{x}) = \mathrm{Pool}(\mathbf{h}_L)$, where $\mathbf{h}_L$ is the final hidden state and Pool extracts a single vector (e.g., the last-position token), and $W = U$ is the unembedding matrix.

2.2 Task-agnostic diagnostic measures

The representation-readout decomposition (Section 2.1, Figure 1a) lets us track two sources of learning dynamics separately: **representation learning**, where the encoder ϕ_θ changes to encourage learning generalizable features, and **readout calibration**, where the readout W changes to exploit the current representation for label prediction, independent of whether that representation generalizes. To analyze the effect of these two competing processes, we introduce diagnostic measures applied to the embedding vectors $\phi_\theta(\mathbf{x})$ and the readout matrix W during training.

For a task with C output labels, group inputs by their label $y(\mathbf{x}) \in \{0, \ldots, C-1\}$ as $\mathcal{X}_c := \{\mathbf{x} \in \mathcal{X} : y(\mathbf{x}) = c\}$, and group their embeddings as $\mathcal{M}_c := \{\phi_\theta(\mathbf{x}) : \mathbf{x} \in \mathcal{X}_c\} \subset \mathbb{R}^N$. Although

[4] Bias terms are absorbed into W via an appended constant feature, so we write $W\phi_\theta(\mathbf{x})$ rather than $W\phi_\theta(\mathbf{x}) + b$. Also, post-readout non-linearity (e.g., softmax) is omitted for simplicity and would not affect the analysis.

$\mathcal{M}_c$ lives in $\mathbb{R}^N$, an extensive literature in machine learning [28, 29, 30, 31, 32] and computational neuroscience [33, 34] reports that these sets are highly structured and locally low-dimensional, and $\mathcal{M}_c$ is often referred to as the *representation manifold* conditioned on label c.[5]

We emphasize that the **diagnostic measures** introduced below are simultaneously task-relevant and task-agnostic. They are task-relevant in that they operate on the label-conditioned manifolds $\{\mathcal{M}_c\}_{c=1}^{C}$. They are task-agnostic in that they exploit no further structure—such as the algebraic properties in modular arithmetic [9] (see Appendix E for preliminary task-specific analyses)—of the task. We view this as a starting point, and defer a systematic extension to future work. See Figure 2 and below for an overview of these measures and Appendix A for detailed definitions and interpretations.

Measure 1 (Critical dimension N_{crit}). The critical dimension N_{crit} is defined as the expected smallest projection dimension preserving separability [31, 14] and is related to manifold capacity theory [15, 35, 14]. Roughly speaking, N_{crit} is the smallest dimensionality N_{proj} such that the label-conditioned manifolds $\{\mathcal{M}_c\}$ are linearly separable with probability at least 0.5 after being randomly projected to an N_{proj}-dimensional subspace (Figure 2b, definitions in Section A.1.1). A small N_{crit} indicates that the representation manifolds are well *untangled* [36, 14]—their linear separability survives aggressive random projection (Figure 9). Crucially, N_{crit} quantifies the robustness of linear separability in that a smaller value indicates a more generalizable representation [11]. See Section A.2 for details

Measure 2 (Task-relevant manifold geometric measures D, R, ρ_c, ρ_a). The GLUE theory [14] also derived a small set of geometric quantities that are analytically linked to the value of N_{crit}: the *effective manifold dimension* D, *effective manifold radius* R, and the center and axis *manifold alignment* ρ_c, ρ_a between pairs of representational manifolds (Figure 2c, definitions in Section A.1.2). The advantage of these GLUE geometric measures is that they are analytically linked to N_{crit} and can better track the task-relevant geometric changes underlying mechanisms of representational robustness [14, 11, 37]. As the definitions of these geometric measures involve an elaborate introduction of the concept of anchor point distribution [14], we defer the details to Section A.3. Previous work using GLUE geometric measures to better understand models' behaviors is discussed in Section C.5.

Measure 3 (Linear probe accuracy). We fit a regularized linear probe [16] on $\phi_\theta(\mathbf{x})$ using the same train/test split as the model and compare its accuracy to the model's. The probe identifies the best regularized linear readout on top of the current representation, providing a reference point that isolates encoder quality from the actual readout: a mismatch between probe and model accuracy signals that the readout is mis-calibrated relative to what ϕ_θ affords (Figure 2d). See Section A.4 for details.

Measure 4 (Last-layer NTK-label alignment). We probe how well the current feature space supports linear readout learning via the alignment between the last-layer neural tangent kernel (NTK) [26] and the label kernel (i.e., similarity matrix of the labels, see definitions in Section A.1.4). This measure, abbreviated NTK alignment, quantifies how closely the feature geometry mirrors the label structure from the readout's perspective: higher alignment indicates that gradient flow on W makes faster progress toward label-aligned solutions (Figure 2e, Section A.5.2). We evaluate alignment on both train and test splits at each checkpoint, allowing us to compare how the feature geometry develops on seen versus unseen data. See more details in Section A.5.

In summary, N_{crit} serves as a scalar summary of representational robustness, and the GLUE geometric measures provide a finer-grained probe of the geometric mechanisms supporting separability. The gap between linear probe and model accuracy diagnoses readout overfitting or underfitting, while the last-layer NTK-label alignment measures how effectively the current feature geometry directs readout gradients toward label-discriminative solutions (Figure 2a).

2.3 Tasks and models

For grokking, there are three algorithmic tasks with known mechanistic solutions: modular addition [2, 9, 18], permutation composition [19, 20], and sparse parity [8, 21], which allow us to test our analysis framework for a task-agnostic explanation for grokking. For each task, we train both an MLP

[5] Also known as object manifolds, neural manifolds, or category manifolds.

and a transformer where applicable. Full data, optimizer, and hyperparameter details are deferred to Appendix B. Below, we briefly specify the specifications of each task.

Modular addition. Predict $(a + b) \bmod p$ for $p = 113$, given a and b as inputs ($C = 113$ classes).

Permutation composition. Predict $\sigma \circ \tau$ in the symmetric group S_n for $n = 5$, given the indices of σ and τ as inputs ($C = 120$ classes).

Sparse parity. Predict the parity of a fixed unknown subset of $k = 3$ coordinates of an input $x \in \{-1, +1\}^n$ with $n = 40$ ($C = 2$ classes).

We also study a fourth grokking example on the small-sample MNIST setup of Liu et al. [3], whose delayed-generalization phenomenon the framework will diagnose as a readout artifact (Section 4.1).

3 Hidden Representation Learning and Biased Readout Under Grokking

We argue that grokking reflects two speeds of learning operating simultaneously: a slow, continuous improvement in representation quality (Section 3.1), and a faster readout calibration that outpaces the encoder and locks onto training data before generalizable representations are ready (Section 3.2).

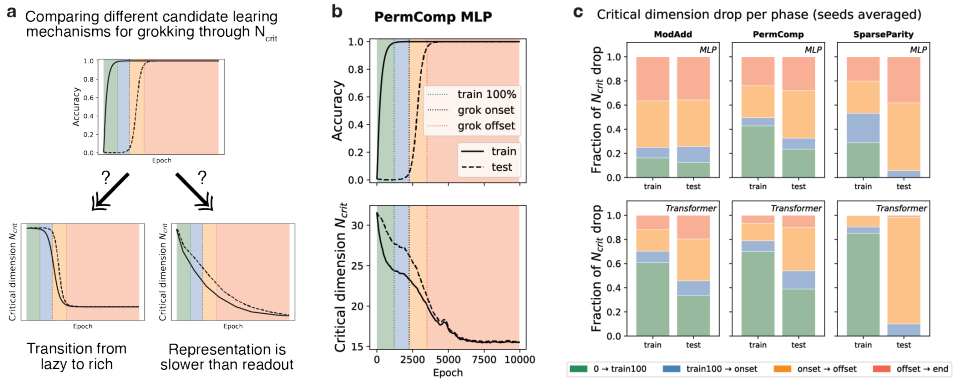

Figure 3: **Representation learning is slow but not absent before grokking. a**, Using critical dimension N_{crit} to compare the candidate mechanism of lazy-to-rich transition (bottom left) and slower-representation-learning (bottom right). **b**, A grokking example. Top: model's accuracy curves. Bottom: N_{crit} decreases throughout the training. **c**, Results across tasks and architectures (averaged over 3 seeds, all results in Section D.1). See Section D.4 for the definition of the shaded areas.

3.1 Representation learning is slow but not absent before grokking

The lazy-to-rich account [10] (Section C.1) attributes grokking to a transition from a *lazy* regime—where features are approximately fixed and the readout does the heavy lifting—to a *rich* regime—where features actively reorganize for better generalization. Under a strict reading of this account, representation learning should not begin until grokking onset. Consequently, critical dimension N_{crit}—which quantifies the degree of representation learning [11] (Section A.2)—should stay near its initial value during early epochs and drop sharply at grokking onset (Figure 3a, bottom left). In contrast, if representation learning were slower than readout calibration but still continuously active in early training, then critical dimension N_{crit} would already decrease before grokking onset (Figure 3a, bottom right).

We compared these two candidate mechanistic explanations by empirically measuring N_{crit} throughout training. Take permutation composition on MLP as an example (Figure 3b), we first color the epochs into four stages according to the model's accuracy curves: (i) before training accuracy reaches 100%, (ii) before grokking onset, (iii) before grokking offset, and (iv) after grokking offset. We found that N_{crit} decreases gradually and continuously across all four stages (Figure 3b, bottom).

To quantify this pattern across tasks and architectures, we report in Figure 3c the fraction of the total N_{crit} drop accounted for by each phase, averaged over three seeds. In every (task, model) pair,

a substantial portion of the drop occurs before grokking onset (i.e., in the green and blue phases combined), ranging from roughly 20%-80% (train) and 5%-50% (test). These findings suggest that representation learning does not begin abruptly at grokking onset—as the strict lazy-to-rich account would predict—rather, it proceeds continuously throughout training. For further discussion on our theory's relation to the lazy-to-rich account, see Section C.1.

3.2 Readout learning direction is train-biased before grokking

Having established that representation learning proceeds continuously throughout training, we now turn to the readout. While the encoder gradually refines the geometry of its representations, the readout has to map those evolving representations to correct outputs for both train and test inputs. A natural question is whether the readout's learning direction is balanced across train and test (generalization), or whether it becomes systematically biased toward the train side (memorization).

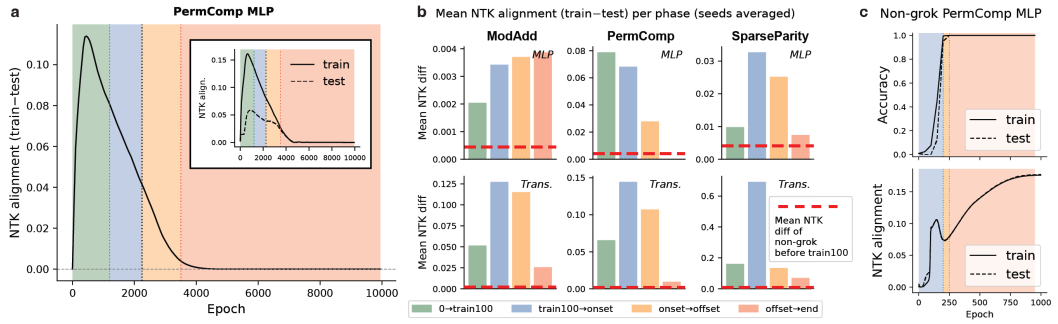

Figure 4: **Readout learning direction is train-biased before grokking. a**, NTK alignment difference (train minus test) throughout training for permutation composition on MLP (seed=0). The gap peaks near training saturation and decays to zero as grokking completes. The inset shows raw train (solid) and test (dashed) NTK alignment curves. See Figure 3a for the model's accuracy curves. **b**, Mean NTK alignment difference per phase. The dashed lines indicate the max difference in the non-grokking counterparts throughout training, as baseline controls. **c**, Accuracy and NTK-label alignment curves for a non-grok PermComp MLP example. See Section D.4 for the definition of the shaded areas.

To probe this, we track the NTK alignment difference (train minus test) throughout training. A positive value indicates that the readout is being updated in a direction that preferentially fits training examples over test examples—a signature of overfitting at the readout level (see Figure 2e for a schematic illustration to interpret the value of NTK alignment, and see Section A.5 for details). As shown for permutation composition on MLP (Figure 4a), the train-test NTK alignment gap is large and positive throughout the pre-grokking phases (green and blue), peaks around the time training accuracy saturates (i.e., the end of green area), and then decays toward zero as grokking completes.

This pattern holds broadly across tasks and architectures (Figure 4b). In every (task, model) pair, the mean NTK alignment difference is typically largest in the train100-to-onset phase (blue bars), substantially exceeding the pre-train100 phase (green) and decaying through the onset-to-offset (orange) and post-offset (red) phases.

The train-test NTK alignment gap disappears when grokking is removed. As a control, we measure the mean NTK alignment difference in non-grokking counterparts for each setting and find that the gap is substantially smaller (Figure 4b, red dashed lines; Figure 4c), confirming that the train-test NTK alignment gap is specific to the grokking regime. Together with the critical dimension results of Section 3.1, this suggests that grokking reflects not a sudden transition in representation learning, but a gradual equalization of readout direction as representation quality catches up.

3.3 Probing underlying algorithms via geometric mechanisms of representation learning

A common mechanistic interpretability approach is to determine whether a network solves a task via a specific algorithm by identifying underlying algorithmic components [9]. Here we show that studying changes in label-conditioned manifolds offers a complementary, top-down[3] route: the geometric measures (Figure 2c, Section A.3) can probe the underlying mechanisms of representation learning and discriminate distinct algorithms implemented by different architectures on the same task.

We identify architecture-dependent differences in the geometric mechanisms underlying representation learning (Figure 5). According to GLUE theory [14], decreasing center alignment ρ_c and increasing axis alignment ρ_a both improve manifold separability (Figure 5a). Across all grokking tasks (ModAdd and PermComp in Figure 5 and SparseParity in Section D.1), MLPs increase ρ_c (hurting N_{crit}) and decrease ρ_a (improving N_{crit}), while Transformers do the opposite (Figure 5b and c)—suggesting that the two architectures achieve representation learning through qualitatively different geometric pathways. Meanwhile, we find that the manifold radius R and dimension D consistently decrease across tasks in both MLPs and Transformers (Section D.1). Together, these results illustrate how the representation-readout decomposition, combined with geometric measures, can discriminate qualitatively different mechanisms that produce the same surface-level grokking behavior, offering a top-down tool for probing the algorithms implemented by neural networks.

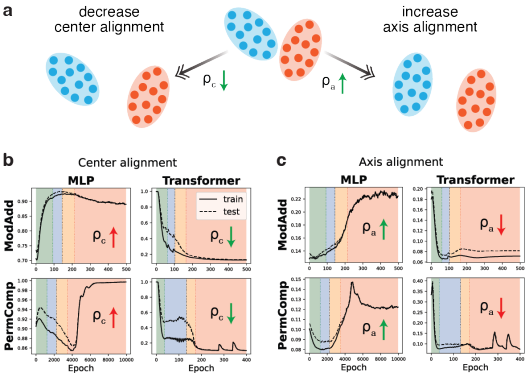

Figure 5: Different geometric mechanisms of representation learning in MLPs and Transformers.

4 Diagnosing Spurious Learning

Both grokking and epoch-wise double descent have served as playgrounds for understanding how neural networks generalize. The clean and extreme forms of these phenomena typically do not arise in real-world training; researchers deliberately induce them via non-standard recipes, such as decreasing weight decay, reducing training data [3], scaling down the relative learning speed of hidden layers [10], or introducing label noise [7]. While studying how to induce these phenomena can be a fruitful approach to building understanding, it can also be misleading if the training recipe strays too far from standard practice. If the ingredients that trigger grokking inadvertently hinder neural network learning, then the scientific conclusions drawn from such settings should be taken with a grain of salt.

We can flag such cases by identifying spurious learning dynamics through the representation-readout decomposition. Specifically, we identify four **diagnostic signatures** as follows.

1. (**Readout overfit**) The model's generalization gap is much larger than that of the LP, indicating that the readout is over-adapting to training data (Figure 6a, top).
2. (**Representation degradation**) Decreasing LP accuracy alongside increasing N_{crit} indicates that representation quality is actively degrading (Figure 6a, bottom).
3. (**Sub-optimal readout**) LP accuracy substantially exceeds model accuracy, indicating that the readout is under-performing relative to what the encoder affords (Figure 7a, top).
4. (**Spurious alignment**) The NTK alignment of a spurious signal substantially exceeds that of the true task signal, indicating that readout gradient updates are increasingly driven by spurious rather than task-relevant structure (Figure 7a, bottom; see also Section A.5).

4.1 Spurious grokking on MNIST via high initial weight norm and small training size

We examine a previously reported grokking example on MNIST [3] (Figure 6). Notably, the final test accuracy in this example is around 80%, while a support vector machine (SVM) can easily achieve near 98% on the same task, suggesting that the training recipe severely limits learning. We use the diagnostic signatures to investigate why. We find that the first two diagnostic signatures appear in this example (Figure 6b). During the pre-grokking epochs (blue region), the model's generalization gap substantially exceeds that of the LP (Figure 6b, double-headed arrows), indicating readout overfitting. Moreover, throughout the pre-grokking period (green and blue regions), LP accuracy decreases and N_{crit} increases (Figure 6b, red boxes), indicating that training is actively degrading representation quality rather than improving it.

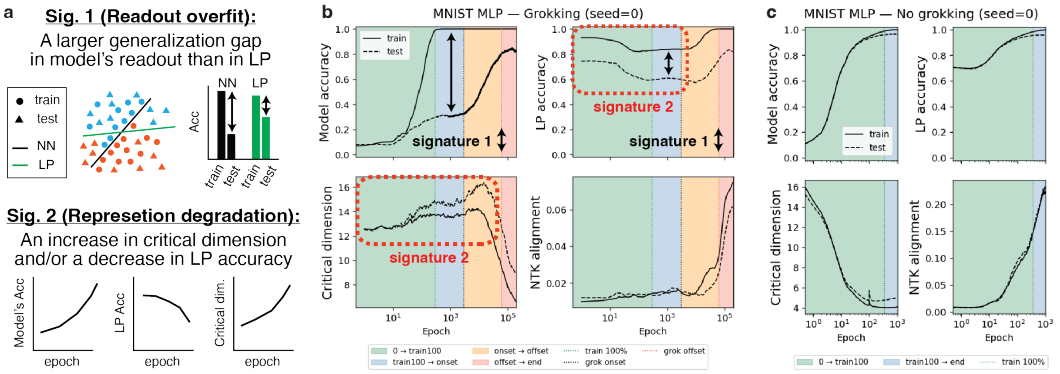

Figure 6: Diagnosing spurious learning in an MNIST grokking example. **a**, Two diagnostic signatures: (i) the generalization gap in the model's readout is much larger than that of a linear probe (top), suggesting that the model overfits; (ii) an increase in critical dimension and/or decrease in linear probe accuracy, suggesting that the feature quality in representations is degrading. **b**, Both diagnostic signatures 1 and 2 appear in the MNIST grokking example reported in [3]. **c**, The diagnostic signatures disappear when we train on MNIST in a standard setup with no grokking. See Section D.4 for the definition of the shaded areas.

Diagnostic signatures disappear under standard training. The grokking recipe of [3] requires multiplying initial weights by a factor greater than four and training on only 1k of the 60k available samples. Removing these non-standard components eliminates both the grokking pattern and the diagnostic signatures (Figure 6c), confirming that the apparent delayed generalization arises from readout miscalibration on top of degrading representations rather than genuine representation improvement.

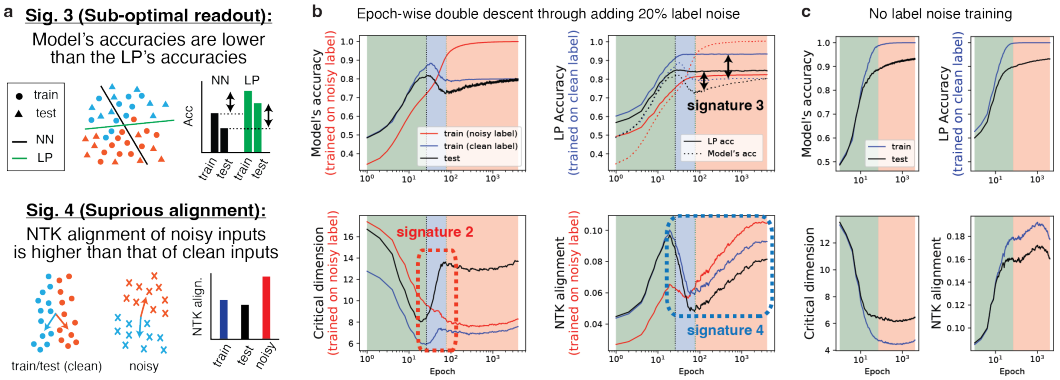

Figure 7: Diagnosing spurious learning in a double descent example. **a**, Two diagnostic signatures: (i) the LP acc (solid lines, upper right) being higher than the model's acc (dashed lines, upper right), (ii) an increase in critical dimension (bottom left), in an epoch-wise double descent example [7]. **b**, Same measures on a 0% label noise example. See Section D.4 for the definition of the shaded areas.

4.2 Epoch-wise double descent through label noise

Epoch-wise double descent is typically induced by introducing label noise [7] (Figure 7). We find that diagnostic signatures 2, 3, and 4 (Figure 7a) appear in this example. LP accuracy on clean labels exceeds the model's accuracy for both train and test (Figure 7b, double-headed arrows), indicating a suboptimal readout. During the test accuracy descent (blue region), LP accuracy plateaus, N_{crit} increases, and NTK alignment decreases (Figure 7b, red boxes), indicating deteriorating representation quality and misaligned readout gradients. In the recovery phase (orange region), the NTK alignment of the noisy-label train set surpasses that of the clean-label train and test sets (Figure 7b, blue boxes), indicating that readout updates become dominated by noisy labels.

Diagnostic signatures disappear when label noise is removed. Removing label noise eliminates both double descent and the diagnostic signatures (Figure 7c), confirming that both arise from the non-standard training recipe. We conclude that epoch-wise double descent here reflects a two-stage process: representation deterioration in the blue phase, followed by spurious signal domination in the orange phase. These results suggest that the diagnostic signatures generalize beyond grokking to non-monotone generalization more broadly, offering a unified lens for diagnosing generalization failures. See Section D.3 for details.

5 Discussion

We proposed a representation-readout decomposition framework (Section 2.1) for studying learning dynamics in deep neural networks, and applied it to understand grokking and epoch-wise double descent. The framework reveals two speeds[2] of learning and yields new insights into the underlying principles (Section 3.1, Section 3.2) and geometric mechanisms (Section 3.3) of these phenomena, and provides diagnostic signatures that distinguish spurious from genuine generalization (Section 4).

The representation-readout decomposition also serves as a top-down[3] complement to existing bottom-up approaches in mechanistic interpretability. Circuit-level analyses excel at identifying specific computational structures, but offer limited guidance across diverse tasks and architectures The decomposition addresses this gap by characterizing how representation quality and readout calibration evolve jointly throughout training, providing principled criteria for where to look and what to expect from circuit-level investigations. We close with several directions for future work.

- **Intermediate-layer analysis.** In this work, the decomposition operates exclusively at the final linear layer. Extending the analysis to intermediate layers is a natural and important direction.
- **Task-specific analysis.** Incorporating task-specific structure into the analysis could reveal more geometric signatures of underlying algorithms (see Appendix E for preliminary findings).
- **Tools for practitioners.** Our diagnostic measures are often consistent across train and test splits (Appendix F), making them reliable training monitors. Developing lightweight estimates and guidance for hyperparameter selection would broaden real-world applicability.
- **Continual and transfer learning.** Investigating encoder stability and readout adaptation in catastrophic forgetting and transfer learning through the lens of our diagnostic signatures.
- **Connections to alignment and safety.** The diagnostic signatures for spurious generalization could provide a principled tool for detecting misalignment in safety-critical models before deployment.

Acknowledgments

We thank members of the Chung lab for helpful discussions and feedback. This work was supported by a Sloan Research Fellowship, a Klingenstein-Simons Award, an Aramont Fellowship in Emerging Science Research, and the Kempner Institute for Natural and Artificial Intelligence at Harvard University. This work was also supported in part by the Center for Computational Neuroscience at the Flatiron Institute, Simons Foundation. All experiments were performed using the Flatiron Institute's high-performance computing cluster. Yao-Yuan Yang served in an advisory capacity.

Code availability

All code required to reproduce the figures presented is available under an MIT License at https://github.com/chung-neuroai-lab/grokking-geometry

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[2] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[3] Ziming Liu, Eric J Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. *arXiv preprint arXiv:2210.01117*, 2022.

[4] Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Deep networks always grok and here is why. *arXiv preprint arXiv:2402.15555*, 2024.

[5] Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization, 2024.

[6] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[7] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[8] Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

[9] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability, 2023. *URL https://arxiv. org/abs/2301.05217*, 2023.

[10] Tanishq Kumar, Blake Bordelon, Samuel J Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *The twelfth international conference on learning representations*, 2023.

[11] Chi-Ning Chou, Hang Le, Yichen Wang, and SueYeon Chung. Feature learning beyond the lazy-rich dichotomy: Insights from representational geometry. In *Forty-second International Conference on Machine Learning*, 2025.

[12] Artem Kirsanov, Chi-Ning Chou, Kyunghyun Cho, and SueYeon Chung. The geometry of prompting: Unveiling distinct mechanisms of task adaptation in language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1855–1888. Association for Computational Linguistics, April 2025.

[13] Xingyu Zheng, Kyle Daruwalla, Ari S Benjamin, and David Klindt. Delays in generalization match delayed changes in representational geometry. In *UniReps: 2nd Edition of the Workshop on Unifying Representations in Neural Models*, 2024.

[14] Chi-Ning Chou, Royoung Kim, Luke Arend, Yao-Yuan Yang, Brett Mensh, Won Mok Shim, Matthew Perich, and SueYeon Chung. Geometry linked to untangling efficiency reveals structure and computation in neural populations. *bioRxiv*, 2025.

[15] SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. Classification and geometry of general perceptual manifolds. *Physical Review X*, 2018.

[16] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*, 2016.

[17] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

[18] Andrey Gromov. Grokking modular arithmetic, 2023.

[19] Bilal Chughtai, Lawrence Chan, and Neel Nanda. A toy model of universality: Reverse engineering how networks learn group operations, 2023.

[20] Dashiell Stander, Qinan Yu, Honglu Fan, and Stella Biderman. Grokking group multiplication with cosets, 2024.

[21] William Merrill, Nikolaos Tsilivis, and Aman Shukla. A tale of two circuits: Grokking as competition of sparse and dense subnetworks. *arXiv preprint arXiv:2303.11873*, 2023.

[22] Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency. *arXiv preprint arXiv:2309.02390*, 2023.

[23] Branton DeMoss, Silvia Sapora, Jakob Foerster, Nick Hawes, and Ingmar Posner. The complexity dynamics of grokking. *arXiv preprint arXiv:2412.09810*, 2024.

[24] Cory Stephenson and Tyler Lee. When and how epochwise double descent happens. *arXiv preprint arXiv:2108.12006*, 2021.

[25] Satvik Golechha. Progress measures for grokking on real-world tasks. *arXiv preprint arXiv:2405.12755*, 2024.

[26] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks, 2020.

[27] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

[28] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[29] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[30] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[31] Uri Cohen, SueYeon Chung, Daniel D Lee, and Haim Sompolinsky. Separability and geometry of object manifolds in deep neural networks. *Nature communications*, 2020.

[32] Lucrezia Valeriani, Diego Doimo, Francesca Cuturello, Alessandro Laio, Alessio Ansuini, and Alberto Cazzaniga. The geometry of hidden representations of large transformer models. *Advances in Neural Information Processing Systems*, 36:51234–51252, 2023.

[33] John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509, 2014.

[34] SueYeon Chung and Larry F Abbott. Neural population geometry: An approach for understanding biological and artificial neural networks. *Current opinion in neurobiology*, 70:137–144, 2021.

[35] Albert J Wakhloo, Tamara J Sussman, and SueYeon Chung. Linear classification of neural manifolds with correlated variability. *Physical Review Letters*, 2023.

[36] James J DiCarlo and David D Cox. Untangling invariant object recognition. *Trends in cognitive sciences*, 11(8):333–341, 2007.

[37] Chi-Ning Chou, Artem Kirsanov, Yao-Yuan Yang, and SueYeon Chung. Diagnosing generalization failures from representational geometry markers. In *The Fourteenth International Conference on Learning Representations*, 2026.

[38] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, pages 326–334, 1965.

[39] Francesca Mignacco, Chi-Ning Chou, and SueYeon Chung. Nonlinear classification of neural manifolds with contextual information. *Physical Review E*, 111(3):035302, 2025.

[40] Cory Stephenson, Suchismita Padhy, Abhinav Ganesh, Yue Hui, Hanlin Tang, and SueYeon Chung. On the geometry of generalization and memorization in deep neural networks. *arXiv preprint arXiv:2105.14602*, 2021.

[41] Nihaad Paraouty, Justin D Yao, Léo Varnet, Chi-Ning Chou, SueYeon Chung, and Dan H Sanes. Sensory cortex plasticity supports auditory social learning. *Nature communications*, 14(1):5828, 2023.

[42] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[43] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of machine learning research*, 3(Nov):463–482, 2002.

[44] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[45] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828, 2012.

[46] Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pages 2269–2277. PMLR, 2021.

[47] Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature communications*, 12(1):2914, 2021.

[48] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[49] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[50] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[51] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[52] Kenzo Clauw, Sebastiano Stramaglia, and Daniele Marinazzo. Information-theoretic progress measures reveal grokking is an emergent phase transition. *arXiv preprint arXiv:2408.08944*, 2024.

[53] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning, 2017.

[54] Vaishnavh Nagarajan and J. Zico Kolter. Generalization in deep networks: The role of distance from initialization, 2019.

[55] Peter Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks, 2017.

[56] Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35:34651–34663, 2022.

[57] Will Slatton, Chi-Ning Chou, and SueYeon Chung. Linear readout of neural manifolds with continuous variables. *arXiv preprint arXiv:2603.10956*, 2026.

[58] Michael Kuoch, Chi-Ning Chou, Nikhil Parthasarathy, Joel Dapello, James J DiCarlo, Haim Sompolinsky, and SueYeon Chung. Probing biological and artificial neural networks with task-dependent neural manifolds. In *Conference on Parsimony and Learning (Proceedings Track)*, 2024.

[59] Bo Hu, Nesibe Z Temiz, Chi-Ning Chou, Peter Rupprecht, Claire Meissner-Bernard, Benjamin Titze, SueYeon Chung, and Rainer W Friedrich. Representational learning by optimization of neural manifolds in an olfactory memory network. *bioRxiv*, pages 2024–11, 2024.

[60] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer Texts in Statistics. Springer, Cham, 5 edition, 2025.

[61] Grace Wahba. Estimating the smoothing parameter. In *Spline Models for Observational Data*, pages 45–65. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[62] Trevor Hastie, Jerome Friedman, and Robert Tibshirani. Smoothing splines. In *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, pages 241–249. Springer, New York, NY, 2017.

Appendix

This appendix provides supplementary material organized as follows.

- Appendix A gives full implementation details and extended theoretical background for the four diagnostic measures used in our analysis: critical dimension, GLUE geometric measures, linear probing, and last-layer NTK-label alignment.
- Appendix B describes the architectures, tasks, and training hyperparameters for all grokking and double descent experiments.
- Appendix C situates our framework within the broader literature, with extended discussions of the lazy-to-rich hypothesis, task-specific progress measures, other task-agnostic measures, weight-based theories of grokking, and previous applications of GLUE theory.
- Appendix D provides additional empirical results for all task-model pairs in both grokking and non-grokking settings, as well as the double descent experiments.
- Appendix E presents preliminary task-specific analyses using pairwise critical dimension heatmaps and transfer linear probes.
- Appendix F quantifies the train-test consistency of our diagnostic measures across tasks, architectures, and training stages, showing that representation-based measures track learning dynamics more reliably than standard loss and accuracy curves.

A Diagnostic Measures for Representation-Readout Decomposition Analysis

In Section 2 and Figure 2, we briefly mention the high-level definition and intuition of the four types of measures used in our analysis. In this appendix section, we are going to provide concrete implementation details (Section A.1), a more in-depth introduction to each method (Section A.2, Section A.3, Section A.4, and Section A.5), and discussions on what other possible measures can be used in future research.

A.1 Implementations of all measures

We begin by providing implementation details for all the measures used in this paper for reproduction. For more contexts, motivations, and theories, please go to the corresponding subsections later.

First, all our measures only depend on the last-layer label-conditioned representations. Recall the notations from Section 2.2: for a task with C output labels, group inputs by their label $y(\mathbf{x}) \in \{0, \dots, C-1\}$ as $\mathcal{X}_c := \{\mathbf{x} \in \mathcal{X} : y(\mathbf{x}) = c\}$, and group their embeddings by the encoder ϕ_θ as $\mathcal{M}_c := \{\phi_\theta(\mathbf{x}) : \mathbf{x} \in \mathcal{X}_c\} \subset \mathbb{R}^N$. Following the convention in recent literature, $\mathcal{M}_c$ is referred to as the *representation manifold* conditioned on label c. See Appendix B for details on the encoder mapping and labels for each task and architecture used in our analysis.

A.1.1 Critical dimension

Consider a collection of dichotomies over C output labels, i.e., a subset $\mathcal{Y}$ in $\{1, -1, 0\}^C$, the critical dimension is defined as follows. When $y_c = 0$, it means that the c-th label does not participate in the dichotomy. Mathematically, assume[6] that $\{\mathcal{M}_c\}$ is linearly separable with respect to every $\mathbf{y} \in \mathcal{Y}$. For $1 \le N_{\mathsf{proj}} \le N$, let $\Pi_{N_{\mathsf{proj}}}$ be a random projection from $\mathbb{R}^N$ to $\mathbb{R}^{N_{\mathsf{proj}}}$, and let $p(N_{\mathsf{proj}})$ denote the probability that $\{\Pi_{N_{\mathsf{proj}}}(\mathcal{M}_c)\}$ remains linearly separable, taken over both the projection draw and a uniform draw of y from $\mathcal{Y}$. By construction, $p(N) = 1$ and $p(1) = 0$. Now, define $N_{\mathsf{crit}} := \sum_{N_{\mathsf{proj}}=1}^{N} \big(1 - p(N_{\mathsf{proj}})\big)$. When $p(\cdot)$ exhibits a symmetric phase transition, N_{crit} approximately coincides with the median-threshold dimension $\arg\max_{N_{\mathsf{proj}}}\{N_{\mathsf{proj}} : p(N_{\mathsf{proj}}) \ge 0.5\}$.

[6] Empirically, this is achieved by downsampling the points in each $\mathcal{M}_c$ to a fixed budget per class and, where indicated, applying a simple kernel.

Numerically, N_{crit} can be estimated by binary search over the value of $p(N_{\mathsf{proj}})$. A better (faster and more accurate) way is using an analytical formula derived in the GLUE theory [14]:

$$N_{\mathsf{crit}} = \mathop{\mathbb{E}}_{\substack{\mathbf{t}\sim\mathcal{N}(0,I_N)\\ \mathbf{y}\sim\mathcal{Y}}} \left[\min_{\substack{\mathbf{v}\in\mathbb{R}^N:\\ y_c\cdot\langle\mathbf{v},\mathbf{z}_c\rangle\geq 0\\ \forall\mathbf{z}_c\in\mathcal{M}_c,\forall c}} \|\mathbf{v}-\mathbf{t}\|_2^2 \right] \tag{2}$$

where $\mathcal{N}(0, I_N)$ denotes the N-dimensional isotropic Gaussian distribution, $\mathbf{y} \sim \mathcal{Y}$ denotes uniformly sampling $\mathbf{y}$ from $\mathcal{Y}$ at random, $\langle\cdot,\cdot\rangle$ denotes inner product between two vectors, and $\|\cdot\|_2$ denotes the ℓ_2 norm of a vector. Note that the minimization inside the expectation is a quadratic program with linear constraints and hence can be efficiently solved by a standard convex optimization solver.

In all our analyses, we sampled 200 random $\mathbf{t}$ to and picked $\mathcal{Y}$ to be the set of all pairwise dichotomies, and randomly subsampled a subset of 30 points from each label-conditioned manifold to ensure linear separability. See Appendix D for results on other choices of dichotomy structure.

A.1.2 GLUE geometric measures

It turns out that through the strong duality property in convex optimization, the formula for the critical dimension in Equation 2 can be rewritten as follows.

$$N_{\mathsf{crit}} = \mathop{\mathbb{E}}_{\substack{\mathbf{t}\sim\mathcal{N}(0,I_N)\\ y\sim\mathcal{Y}}} \left[\max_{\substack{\mathbf{s}_c\in\mathsf{Hull}(\mathcal{M}_c)\\ \lambda_c\geq 0}} \left(\frac{\langle \mathbf{t}, \sum_c y_c\lambda_c\mathbf{s}_c\rangle}{\|\sum_c y_c\lambda_c\mathbf{s}_c\|_2} \right)^2 \right] \tag{3}$$

where $\mathsf{Hull}(\mathcal{M}_c)$ denotes the convex hull of $\mathcal{M}_c$, and $\mathsf{span}(\{s_c\})$ denotes the linear subspace spanned by $\{s_c\}$. Note that the maximization inside the expectation is simply the square of the ℓ_2 norm of the projection of t onto the convex cone spanned by the representation manifolds weighted by their respective labels y.

For each $(\mathbf{y}, \mathbf{t})$ pair sampled by the expectation operator in Equation 3, we can denote $\mathbf{s}_c(\mathbf{y}, \mathbf{t}) \in \mathsf{Hull}(\mathcal{M}_c)$ for each c as the optimizer for the inner maximization term. Namely, the distribution of $(\mathbf{y}, \mathbf{t})$induces a distribution over each $\mathsf{Hull}(\mathcal{M}_c)$, this is also known as the *anchor point distribution* in GLUE theory [14]. Next, for each label c, we can then define the anchor center to be $\mathbf{s}_{c,0} := \mathbb{E}_{\mathbf{y},\mathbf{t}}[\mathbf{s}_c(\mathbf{y}, \mathbf{t})]$. And for each $(\mathbf{y}, \mathbf{t})$ pair, we can define the axis part of the anchor point of the c-th manifold as $\mathbf{s}_{c,1}(\mathbf{y}, \mathbf{t}) = \mathbf{s}_c(\mathbf{y}, \mathbf{t}) - \mathbf{s}_{c,0}$. Finally, denote $\mathbf{S}_0 \in \mathbb{R}^{C\times N}$ as the matrix with the c-th row being $\mathbf{s}_{c,0}$ and $\mathbf{S}_1(\mathbf{y}, \mathbf{t}) \in \mathbb{R}^{C\times N}$ as the matrix with the c-th row being $\mathbf{s}_{c,1}(\mathbf{y}, \mathbf{t})$.

- Effective manifold dimension D is defined as

$$D := \mathop{\mathbb{E}}_{\mathbf{y},\mathbf{t}} \left[\frac{1}{P_{\mathbf{y}}} (\mathbf{S}_1(\mathbf{y}, \mathbf{t})\mathbf{t})^\top (\mathbf{S}_1(\mathbf{y}, \mathbf{t})\mathbf{S}_1(\mathbf{y}, \mathbf{t})^\top)^\dagger (\mathbf{S}_1(\mathbf{y}, \mathbf{t})\mathbf{t}) \right] \tag{4}$$

  where $P_{\mathbf{y}}$ denotes the number of non-zero entries in $\mathbf{y}$ (i.e., number of labels that are in comparison by $\mathbf{y}$), and $\dagger$ denotes matrix pseudoinverse.

- Effective manifold radius R is defined as

$$R := \sqrt{\frac{\mathbb{E}_{\mathbf{y},\mathbf{t}}[c(\mathbf{y}, \mathbf{t})]}{\mathbb{E}_{\mathbf{y},\mathbf{t}}[b(\mathbf{y}, \mathbf{t}) - c(\mathbf{y}, \mathbf{t})]}} \tag{5}$$

  where $b(\mathbf{y}, \mathbf{t}) = (\mathbf{S}_1(\mathbf{y}, \mathbf{t})\mathbf{t})^\top (\mathbf{S}_1(\mathbf{y}, \mathbf{t})\mathbf{S}_1(\mathbf{y}, \mathbf{t})^\top)^\dagger (\mathbf{S}_1(\mathbf{y}, \mathbf{t})\mathbf{t})$ and $c(\mathbf{y}, \mathbf{t}) = (\mathbf{S}_1(\mathbf{y}, \mathbf{t})\mathbf{t})^\top (\mathbf{S}_0\mathbf{S}_0^\top + \mathbf{S}_1(\mathbf{y}, \mathbf{t})\mathbf{S}_1(\mathbf{y}, \mathbf{t})^\top)^\dagger (\mathbf{S}_1(\mathbf{y}, \mathbf{t})\mathbf{t})$.

- Anchor center alignment ρ_c is defined as

$$\rho_c := \frac{1}{C(C-1)} \sum_{c\neq c'} \frac{|\langle \mathbf{s}_{c,0}, \mathbf{s}_{c',0}\rangle|}{\|\mathbf{s}_{c,0}\|_2 \cdot \|\mathbf{s}_{c',0}\|_2}. \tag{6}$$

- Anchor axis alignment ρ_a is defined as

$$\rho_a := \frac{1}{C(C-1)} \sum_{c\neq c'} \mathop{\mathbb{E}}_{\mathbf{y},\mathbf{t}} \left[\frac{|\langle \mathbf{s}_{c,1}(\mathbf{y}, \mathbf{t}), \mathbf{s}_{c',1}(\mathbf{y}, \mathbf{t})\rangle|}{\|\mathbf{s}_{c,1}(\mathbf{y}, \mathbf{t})\|_2 \cdot \|\mathbf{s}_{c',1}(\mathbf{y}, \mathbf{t})\|_2} \right]. \tag{7}$$

In all our analyses, we sampled 200 random $\mathbf{t}$ to estimate N_{crit} and the GLUE geometric measures. Also, we considered random pairwise manifold comparisons, and each manifold contains 30 points randomly sampled from the point cloud. For each configuration, we ran 50 repeats and report the average results. We provide detailed derivations and intuitions for these GLUE geometric measures in Section A.3.

A.1.3 Linear probe

Given C label-conditioned manifolds $\{\mathcal{M}_c\}$ in $\mathbb{R}^N$, a linear probe is a matrix $W \in \mathbb{R}^{C\times N}$ that performs C-class classification through a linear map $\mathbf{x} \mapsto W\phi_\theta(\mathbf{x})$, followed by a softmax, where W is fit by minimizing a regularized cross-entropy loss on the training set while keeping the encoder ϕ_θ frozen.

We fit W using AdamW with learning rate 10^{-3}, $(\beta_1, \beta_2) = (0.9, 0.98)$, and weight decay 1.0 for 200 full-batch epochs. The strong weight decay ensures that high probe accuracy reflects genuinely structured representations rather than memorization. W is initialized as $W_{ij} \sim \mathcal{N}(0, 1/N)$ with no bias term. The train-test split matches that of the model being probed, and we report both train and test probe accuracy. See more details and interpretations in Section A.4

A.1.4 Last-layer NTK-label alignment

The last-layer NTK-label alignment quantifies how well the network's last-layer kernel aligns with the label structure. For an encoder $\phi_\theta : \mathcal{X} \to \mathbb{R}^N$ with a linear readout and a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, the last-layer NTK takes the block-diagonal form $K_{\mathrm{NTK},(i,c),(i',c')} = \delta_{cc'}\,\phi_\theta(\mathbf{x}_i)^\top\phi_\theta(\mathbf{x}_{i'})$, i.e., the feature Gram matrix $K = \Phi\Phi^\top$ replicated once per class. Here $\delta_{cc'}$ denotes the Delta function, i.e., $\delta_{cc'} = 1$ if $c = c'$ and $\delta_{cc'} = 0$ if $c \neq c'$. The alignment is the centered kernel alignment (CKA) between K_{NTK} and the label kernel $Y_{(i,c),(i',c')} = \delta_{cc'}\mathbf{1}[y_i = c]$, defined via the Hilbert-Schmidt Independence Criterion $\mathrm{HSIC}(K, K') = \frac{1}{(n-1)^2}\mathrm{tr}(KHK'H)$, where $H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is the centering matrix:

$$\mathrm{Align}(K_{\mathrm{NTK}}, Y) = \frac{\mathrm{HSIC}(K_{\mathrm{NTK}}, Y)}{\sqrt{\mathrm{HSIC}(K_{\mathrm{NTK}}, K_{\mathrm{NTK}})\,\mathrm{HSIC}(Y, Y)}}\,. \tag{8}$$

Since K_{NTK} factors through the feature Gram matrix, all three HSIC terms reduce to closed-form traces involving K and the centered one-hot label matrix $\tilde{Y} \in \mathbb{R}^{n\times C}$, requiring no backward passes or Hutchinson estimation. We compute alignment on both train and test sets at each checkpoint; see Section A.5 for the full derivation, interpretations, and intuitions.

A.2 Critical dimension

Critical dimension is the key concept in GLUE theory that quantifies the average-case separability—also known as the degree of manifold untangling—of neural representations. It was tightly related to the notion of manifold capacity [15, 35, 14] from theoretical neuroscience and statistical physics (see Section C.5). Recently, Chou & Le et al. [11] showed that critical dimension excels at quantifying the degree of feature learning compared to other previously used kernel-based and weight-based measures.

In this subsection, we first explain the connection between critical dimension and the closely related notion of manifold capacity in Section A.2.1. Next, we provide examples to illustrate why critical dimension captures the average-case separability of neural representations in Section A.2.2. Finally, we explain the connection to feature learning in Section A.2.3.

A.2.1 Connection between critical dimension and manifold capacity

It all began with the concept of perceptron capacity, which asks how many distinct points can be stored in an N-dimensional space such that every dichotomy over these points admits a linear classifier. Cover [38] famously showed that the answer is $2N$ as long as the points are not degenerate (i.e., in general position). Namely, the *capacity* of a neural network for linear readout (i.e., perceptron) is 2.

While the notion of perceptron capacity spurred a rich line of work on figuring out if it's possible to increase capacity of a network, or the capacity of other types of readout, etc., a recent line of work on

manifold capacity questions the beginning assumptions on modeling the patterns of representation by a single point in the neural activity space. The idea is that the collection of neural representations for the same category (e.g., cat or dog) may not collapse into a single point but rather occupy a manifold (which mathematically is simply a convex set, as here the readout is linear) in the neural activity space. So they asked about the capacity of a network for storing manifold-like patterns.

Following the tradition of the statistical physics approach in perceptron capacity, the initial attempt to define manifold capacity is through *mean-field models* [15, 35, 39], which are mathematical models with infinitely many neurons N_{mf} and infinitely many manifolds P_{mf}. In such a limit (i.e., $P_{\mathsf{mf}}, N_{\mathsf{mf}} \to \infty$ with the load $P_{\mathsf{mf}}/N_{\mathsf{mf}}$ holds as a constant, a.k.a., thermodynamic limit), there is a phase transition on the linear separability of the mean-field model with respect to the load $P_{\mathsf{mf}}/N_{\mathsf{mf}}$. Concretely, there exists $\alpha > 0$ such that the mean-field model is linearly separable for all dichotomies if and only if $P_{\mathsf{mf}}/N_{\mathsf{mf}} \leq \alpha$.

While mean-field models are analytically solvable, the downside is that one must make assumptions about how the data manifolds are distributed. For example, no correlations between the manifolds [15] or Gaussian correlations [35]. These assumptions allow some cruel approximations in earlier applications of the manifold capacity theory [31, 40, 41]. However, the mean-field assumptions often do not hold in real-world data (e.g., Figure 2 in [14]). This motivated a data-driven approach to manifold capacity, which was cultivated in the GLUE (Geometry Linked to Untangling Efficiency) theory [14]. The key idea in GLUE is that, instead of building a mathematical model that generates infinitely many manifolds, they directly study a given number of manifolds from data and the dichotomies of interest, and then determine the number of subpopulations of neurons required to support linear separability with respect to these readouts. And this is exactly the notion of critical dimension as formally defined in Section A.1.1. Finally, in GLUE theory, the data-driven version of manifold capacity is hence defined as $P_{\mathsf{data}}/N_{\mathsf{crit}}$ where P_{data} is the number of manifolds in the data.

We encourage the interested reader to read the supplementary materials of [14] for more details on the history of manifold capacity.

A.2.2 Critical dimension quantifies average-case separability

To build intuition for critical dimension, we use a simple two-dimensional toy example that isolates the difference between *best-case* and *average-case* linear separability.

Best-case versus average-case separability. Consider two Gaussian classes in $\mathbb{R}^2$: class $+$ with mean $(\mu, 0)$ and class $-$ with mean $(-\mu, 0)$, both sharing covariance $\Sigma = \mathrm{diag}(\sigma_1^2, \sigma_\perp^2)$. The best-case linear readout—i.e., the optimal linear discriminant (LDA) or the support vector machine (SVM)—recovers the Fisher direction $\Sigma^{-1}(\mu_+ - \mu_-) \propto e_1$ regardless of $\sigma_\perp$. The resulting accuracy is $\Phi(\mu/\sigma_1)$, which is *entirely independent of* $\sigma_\perp$. Similarly, the SVM margin is determined solely by the separation 2μ and σ_1.

A toy with two parallel ellipsoids. We now vary $\sigma_\perp$ while holding μ and σ_1 fixed. Three regimes are illustrated in Figure 8: (i) *needles* ($\sigma_\perp \ll \sigma_1$), where both classes form elongated cigars along x_1; (ii) *circles* ($\sigma_\perp = \sigma_1$), where both classes are isotropic; and (iii) *pancakes* ($\sigma_\perp \gg \sigma_1$), where both classes spread widely along x_2. In all three regimes, the optimal separating hyperplane is $x_1 = 0$ and the best-case accuracy is identical.

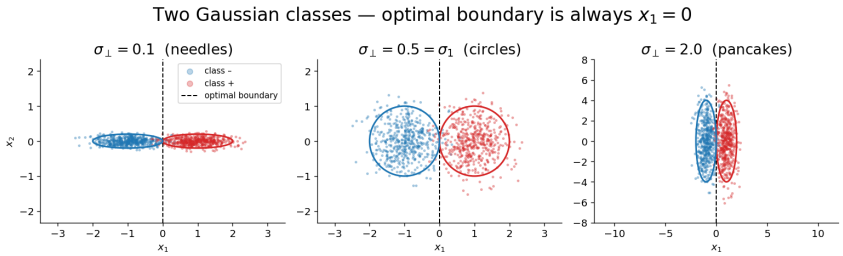

Figure 8: Three regimes of the two-ellipsoid toy. Both classes share the same covariance $\Sigma = \mathrm{diag}(\sigma_1^2, \sigma_\perp^2)$; the optimal separating boundary (dashed) is always $x_1 = 0$ regardless of $\sigma_\perp$.

However, these three regimes differ dramatically in their *average-case* separability. Consider projecting the data onto a random one-dimensional direction $\theta \in [0, \pi)$; Figure 9 illustrates several such projections and the resulting one-dimensional class distributions. For projections near the optimal direction $\theta \approx 0$ the two classes separate cleanly, but for directions near $\theta \approx \pi/2$ the projected distributions overlap strongly—and this overlap worsens as $\sigma_\perp$ grows.

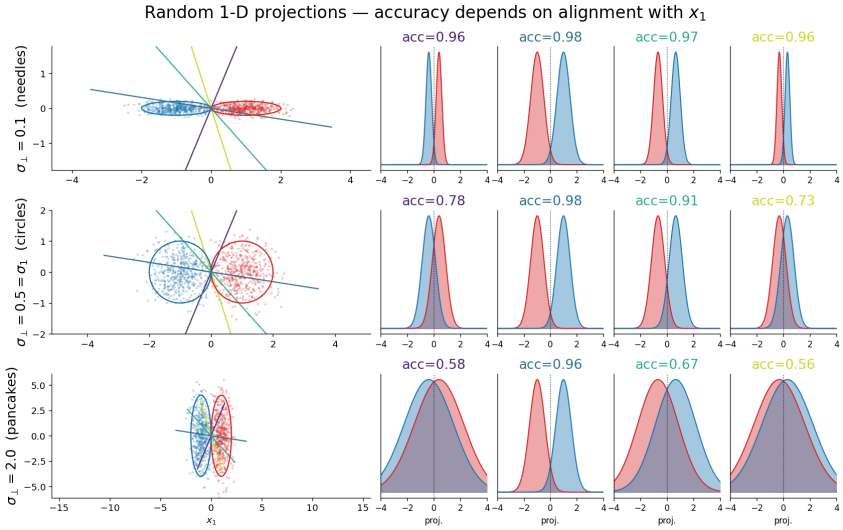

Figure 9: Random 1-D projections of the two-ellipsoid toy (one column per projection direction, one row per regime). Each panel shows the Bayes-optimal accuracy for that projection direction and the resulting one-dimensional class distributions. As $\sigma_\perp$ grows, more projection directions yield near-chance separation.

The Bayes-optimal accuracy for a projection at angle θ is

$$\mathrm{Acc}(\theta) = \Phi\left(\frac{\mu|\cos\theta|}{\sqrt{\sigma_1^2\cos^2\theta + \sigma_\perp^2\sin^2\theta}}\right), \tag{9}$$

where Φ is the standard normal CDF. Figure 10 shows $\mathrm{Acc}(\theta)$ as a polar plot over all projection angles. In the needle regime, the filled region is nearly circular at high accuracy: almost any random projection recovers the class structure. In the pancake regime, the region collapses to a narrow lobe near $\theta = 0$—a *cone of good projections*—while the rest of the circle sits near chance. The mean random-projection accuracy $\frac{1}{\pi}\int_0^\pi \mathrm{Acc}(\theta)\, d\theta$ therefore decreases monotonically with $\sigma_\perp$.

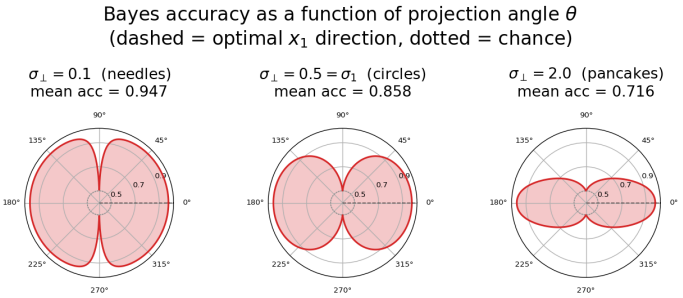

Figure 10: Polar plot of the Bayes-optimal 1-D projection accuracy $\mathrm{Acc}(\theta)$ as a function of projection angle θ for each regime. The filled region shrinks from a broad disc (needles) to a narrow lobe (pancakes), visualizing the shrinking cone of good projections. Dotted circle marks chance accuracy (0.5).

Critical dimension tracks average-case separability. Figure 11 summarizes all four quantities as $\sigma_\perp$ is swept over two orders of magnitude. The best-case measures—optimal LDA accuracy and LinearSVC accuracy—remain flat throughout, confirming that the optimal readout is blind to $\sigma_\perp$. In contrast, the mean random-projection accuracy (middle panel) decreases monotonically, and the GLUE critical dimension (right panel) increases in lockstep.

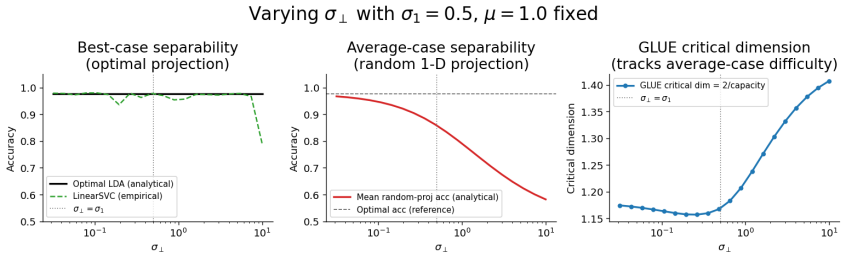

Figure 11: Summary sweep over $\sigma_\perp$. *Left*: best-case separability (optimal LDA and LinearSVC) is flat. *Middle*: mean random-projection accuracy decreases monotonically. *Right*: Critical dimension increases monotonically, tracking the degradation in average-case separability.

Intuition for neural representations. Why does average-case separability matter? In practice, the downstream decoder that reads out from neural representations does not have access to the globally optimal direction—it operates on a finite training set and must generalize. Moreover, a single population of neurons may need to support multiple distinct readouts simultaneously, only a few of which are aligned with any particular optimal direction. When the critical dimension is low, weakly optimized or randomly initialized readout directions are already informative, and the representation is said to be *untangled*. When the critical dimension is high, only a narrow subset of directions supports accurate decoding, so downstream decoders must work harder to identify the correct subspace. This is precisely the sense in which critical dimension quantifies the *degree of manifold untangling* referenced in earlier work [36, 15, 14].

A.2.3 Critical dimension quantifies the degree of feature learning

Chou and Le et al. [11] proposed to study feature learning through the geometry of task-relevant manifolds, where a task-relevant manifold is the point cloud of neural activity patterns corresponding to a given category or condition. Their central observation is that feature learning can be viewed as a process of *untangling* task-relevant manifolds—structuring the neural representational space to improve separation among manifolds. Concretely, in a network that does not learn task-relevant features (e.g., the lazy or random-feature regime), the manifolds are poorly organized, making them harder to distinguish (e.g., smaller margin, smaller solution volume). In contrast, when a network learns task-relevant features (e.g., rich learning, neural collapse), the manifolds become well-organized and easier to separate (e.g., larger margin, larger solution volume).

As established in Section A.2.2, manifold capacity α quantifies this degree of untangling via an *average-case* notion of separability, and critical dimension $N_{\text{crit}} = 2/\alpha$ is its reciprocal measure: lower critical dimension means the manifolds are more untangled. This average-case perspective is precisely what makes critical dimension well-suited for quantifying feature learning. As noted by [11], the margin in support vector machine (SVM) theory quantifies the degree of separability in the worst-case setting, whereas manifold capacity (and hence critical dimension) is average-case in the sense of the random projection—capturing not just whether an optimal readout exists, but how broadly separable the representations are.

Toy setup. To demonstrate this empirically, [11] considers 2-layer non-linear networks of the form $f(\mathbf{x}) = \frac{1}{\sqrt{N}}\mathbf{a}^\top \sigma(W^\top \mathbf{x})$, where $W \in \mathbb{R}^{N\times d}$ is the hidden-layer weight matrix, $\mathbf{a} \in \mathbb{R}^N$ is a fixed random readout, and σ is a ReLU activation. The networks are trained on Gaussian cloud data in a teacher–student setting, where the input–label pairs are generated by a hidden signal β^*. The inverse scale factor $\bar{\eta} \in \{2^0, 2^1, 2^2, 2^3, 2^4\}$ controls the effective learning rate, interpolating from a lazy (kernel-like) regime at $\bar{\eta} = 2^0$ to a rich (feature-learning) regime at $\bar{\eta} = 2^4$. We reproduce their experiment and compare five measures across all richness levels: critical dimension,

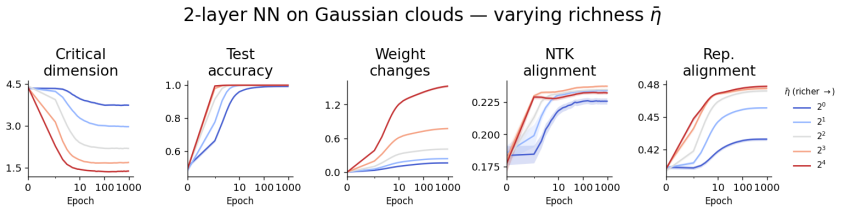

Figure 12: Critical dimension tracks the degree of feature learning. 2-layer non-linear networks are trained on Gaussian clouds with the inverse scale factor $\bar{\eta}$ varied across five richness levels (blue 2^0 laziest to red 2^4 richest; shading is ± 1 SEM over seeds). Critical dimension $N_{\text{crit}} = 2/\alpha$ (left panel) correctly orders all five conditions throughout training: richer $\bar{\eta}$ yields lower critical dimension. Test accuracy (second panel) is insensitive to richness; weight changes (third), NTK–label alignment (fourth), and representation–label alignment (fifth) exhibit incorrect orderings in some regimes. Experiment reproduced from [11].

test accuracy, weight changes $\|W_t - W_0\|_F / \|W_0\|_F$, NTK–label alignment $\text{CKA}(K_t^{\text{NTK}}, \mathbf{y}\mathbf{y}^\top)$, and representation–label alignment $\text{CKA}(X_t X_t^\top, \mathbf{y}\mathbf{y}^\top)$.

Results. Figure 12 shows the result. Critical dimension correctly and consistently orders all five richness levels throughout training: richer training (higher $\bar{\eta}$) produces representations with lower critical dimension, i.e., more untangled manifolds, as expected. The other measures fail in different ways. Test accuracy is nearly identical across all conditions and cannot distinguish richness levels at any point during training. Weight changes and NTK–label alignment track some differences early in training but exhibit crossings and incorrect orderings in later epochs. Representation–label alignment shows the wrong ordering entirely in some regimes, consistent with the finding in [11] that the representation-label alignment would characterize the wrong ordering of wealthiness in initial features. As the authors conclude, manifold capacity excels at extracting task-relevant structures in representations because it is data-driven and free from additional statistical assumptions on the data.

Theoretical grounding. [11] also provide formal justification for this finding via Theorem 3.1, which states that in the proportional asymptotic limit ($P_{\text{train}}, d, N \to \infty$ at the same rate), capacity monotonically tracks the learning rate: $\alpha(\eta, \psi_1, \psi_2) < \alpha(\eta', \psi_1, \psi_2)$ for every $0 < \eta < \eta'$. Moreover, there exists an increasing invertible function $h_{\psi_1,\psi_2} : \mathbb{R}_+ \to [0, 1]$ such that prediction accuracy satisfies $\text{Acc}(\eta, \psi_1, \psi_2) = h_{\psi_1,\psi_2}(\alpha(\eta, \psi_1, \psi_2))$, establishing a monotone link between capacity and accuracy. Translating to critical dimension: lower N_{crit} implies higher capacity, which implies higher prediction accuracy—making critical dimension a principled, theoretically grounded measure of the degree of feature learning.

A.3 GLUE geometric measures

In Section A.2 we introduced the connection between critical dimension and manifold capacity theory. One thing we skipped was about how the geometric organization of manifolds relates to the value of manifold capacity. Early work of Chung et al. [15] used the mean-field formula for manifold capacity to identify that manifold dimension and manifold radius contribute to the capacity value. Specifically, the smaller the manifold dimension and radius, the higher the manifold capacity. One important thing is the anchor point distribution, which better captures the task-relevant structure of data, analogous to the support vector in SVM, but now it's average-case as opposed to the best-case in SVM.

The original mean-field theory framework only captured *intra-class* geometry through dimension and radius. The GLUE framework [14] extends this by introducing two additional measures that capture *inter-class* geometric structure: anchor center alignment ρ_c and anchor axis alignment ρ_a. Center alignment measures how similarly the anchor centers of different classes are oriented in representation space — when ρ_c is high, the mean anchor points of different classes point in similar directions, making them harder to separate. Axis alignment measures how parallel the local variation directions are across classes — when ρ_a is high, the per-sample fluctuations of different manifolds are correlated, which reduces the effective diversity of random projections and thus lowers capacity. Formal definitions of all four measures are given in Section A.1.2.

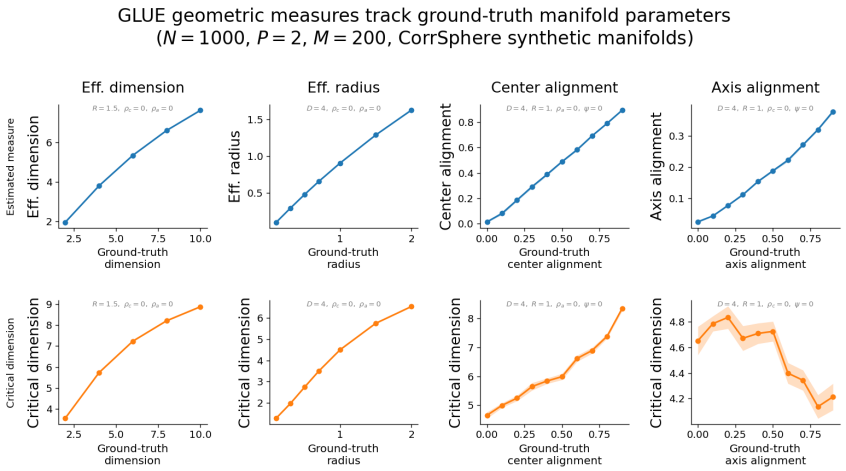

Figure 13: **Intuition for GLUE geometric measures.** Each measure is validated on synthetic Gaussian manifolds ($N = 1000$, $P = 2$, $M = 200$) from [14]. Each column sweeps one ground-truth parameter while holding the others fixed. **Top:** estimated GLUE measure vs. ground-truth parameter. **Bottom:** critical dimension $N_{\text{crit}} = 2/\alpha$, where α is manifold capacity, vs. ground-truth parameter.

Together, these four measures decompose manifold geometry into two orthogonal aspects: intra-class geometry (D, R), which governs how spread out each manifold is on its own, and inter-class geometry (ρ_c, ρ_a), which governs how geometrically similar different class manifolds are to one another. In all cases, values that indicate more "complex" or entangled geometry — larger dimension, larger radius, higher center alignment, higher axis alignment — correspond to lower manifold capacity and thus higher critical dimension. This decomposition provides an interpretable diagnostic: when we observe a drop in critical dimension during training, we can attribute it to specific geometric changes in the learned representations.

To build intuition for these four measures, we reproduce the validation experiment from [14] in Figure 13. The toy example generates synthetic Gaussian manifolds where each ground-truth parameter is swept independently, serving as a controlled setting where the geometric meaning of each measure can be verified directly. The top row confirms that each GLUE estimate faithfully tracks its corresponding ground-truth parameter; the bottom row shows that critical dimension inherits a monotone relationship with each, validating it as a composite readout of all four geometric effects.

A.4 Linear probe

Linear probing [16] is a diagnostic tool for evaluating the quality of learned representations. A linear classifier is trained on top of frozen network activations, and its accuracy measures whether class-relevant structure is *linearly accessible* in the representation—decoupled from whether the full network exploits it via nonlinear layers. While prior work has used layer-wise LP accuracy to track how information propagates through the network [16], we focus exclusively on the final-layer representation.

In analytically tractable settings, LP accuracy has a precise geometric meaning. In the SVM framework [42], LP accuracy tracks linear separability and the margin directly quantifies the degree of separation [43]. In kernel and ridge regression [44], generalization is governed by the linear separability of the kernel features, so LP accuracy on those features directly bounds the generalization gap. In the NTK / lazy training regime [17], the network behaves as a kernel machine throughout training, and LP accuracy on the final layer is equivalent to the kernel classifier—LP accuracy and test accuracy coincide with no gap between them.

In deep networks trained in the rich / feature-learning regime, this equivalence breaks down. The LP classifier is applied directly to the frozen final-layer pre-logits, which are already the input to a linear output head, so LP involves no hidden layer. However, the nonlinear layers upstream can produce final-layer representations with poor linear structure even when train accuracy is high—the network routes

around this via its nonlinear composition. LP accuracy exposes this gap: it measures representation quality independently of whether the network's downstream classifier uses the representation correctly.

A.5 Last-layer NTK-label alignment

In this subsection, we offer more details on the (last-layer) NTK-kernel alignment measure as introduced in Section 2.2 and Section A.1.4. Specifically, we provide 2D toy examples in Section A.5.1 to provide more intuitions on thinking about the value of NTK alignment, and a connection between NTK alignment and readout weight learning dynamics in Section A.5.2.

A.5.1 Intuition for last-layer NTK alignment

We refer the reader to Section A.1.4 for the formal definition of last-layer NTK alignment. This section develops the geometric intuition behind the metric through two contrasting binary classification toy examples. By the end, the reader should have a clear picture of what the alignment score measures, why it is sensitive to class-discriminative structure in the representations, and why it rises as a network transitions from lazy to rich learning during grokking.

Binary classification. Consider a dataset of n samples with features $\Phi \in \mathbb{R}^{N\times n}$ and binary labels $y_i \in \{+1, -1\}$. The feature Gram matrix $K = \Phi^\top \Phi \in \mathbb{R}^{n\times n}$, shown in the *middle panel* of Figure 14 and Figure 15, records all pairwise inner products $K_{ij} = \phi_i^\top \phi_j$ between representations. The label kernel $Y \in \mathbb{R}^{n\times n}$ where $Y_{i,j} = y_i y_j$, shown in the *right panel*, encodes class structure: $Y_{ij} = +1$ when samples i and j share a class and $Y_{ij} = -1$ otherwise. Alignment between K and Y is their normalized Frobenius inner product,

$$\text{alignment}(K, Y) = \frac{\langle K, Y\rangle_F}{\|K\|_F \|Y\|_F} \propto \sum_{\substack{i,j \\ \text{same class}}} K_{ij} - \sum_{\substack{i,j \\ \text{diff. class}}} K_{ij}. \tag{10}$$

where $\langle K, Y\rangle_F = \sum_{i,j} K_{ij} Y_{ij}$ is the Frobenius inner product and $\|K\|_F = \sqrt{\langle K, K\rangle_F}$ is the Frobenius norm. Equation 10 says that alignment is high precisely when within-class representations are more similar to each other than to representations from the other class (i.e., when the features cluster by label).

This has a direct connection to gradient-based learning. At initialization, with readout weights $W = 0$, the softmax outputs uniform class probabilities $p(c \mid \phi_i) = \frac{1}{2}$ for all samples. The gradient of the cross-entropy loss with respect to the readout weight vector for class c is

$$-\frac{\partial \ell}{\partial w_c}\bigg|_{W=0} \propto \bar{\phi}_c - \bar{\phi}, \tag{11}$$

where w_c is the readout weight vector of class c, and $\bar{\phi}_c$ is the mean representation of class c and $\bar{\phi}$ is the overall mean. For binary classification, $\bar{\phi}_{+1} - \bar{\phi} = \frac{1}{2}(\bar{\phi}_{+1} - \bar{\phi}_{-1})$, so both readout vectors are pushed in opposite directions along the class-mean difference vector, i.e., $\bar{\phi}_{+1} - \bar{\phi}_{-1}$, shown as arrows in the inset of the *left panel*. If this difference is large relative to the within-class spread, the first gradient step is informative and K aligns well with Y. If the class means nearly coincide, the gradient is small and K carries no class structure, giving low alignment. We refer the reader to Section A.5.2 for a more detailed treatment of the connection between readout learning dynamics and NTK alignment.

Example 1: high NTK-label alignment. We sample $N = 300$ points per class from two Gaussians,

$$\phi \sim \begin{cases} \mathcal{N}\big([1,\ 0]^\top,\ \text{diag}(0.5,\ 3.0)\big) & \text{class } +1, \\ \mathcal{N}\big([-1,\ 0]^\top,\ \text{diag}(0.5,\ 3.0)\big) & \text{class } -1. \end{cases} \tag{12}$$

The class means are well-separated (by 2 units along x_1), while both classes share the same large variance along x_2. Because the x_2 variance is identical for both classes, it does not contribute to $\bar{\phi}_{+1} - \bar{\phi}_{-1}$; only the discriminative x_1 direction matters. As visible in the left panel of Figure 14, the inset arrows point cleanly along $\pm x_1$, confirming that the readout gradient is informative. The Gram matrix K (middle panel) displays a clear block structure, with within-class inner products consistently positive and large while between-class ones are negative, and therefore aligns strongly with the label kernel Y (right panel).

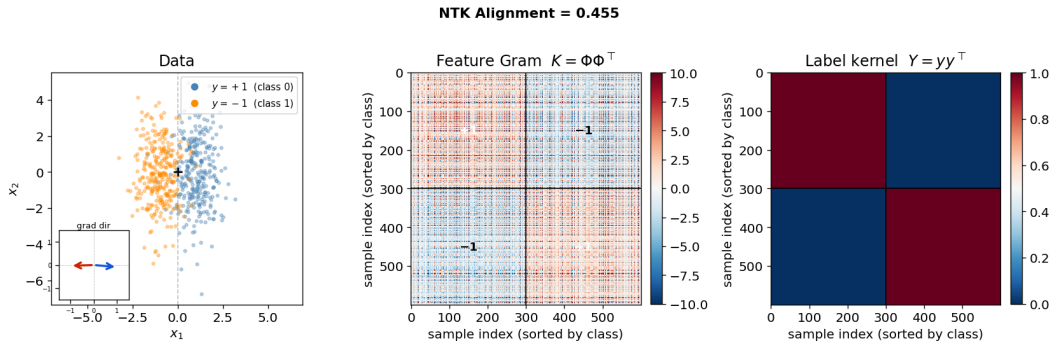

Figure 14: A toy 2D example of high NTK alignment.

Example 2: low NTK-label alignment. We sample the same number of points from

$$\phi \sim \begin{cases} \mathcal{N}\big([0.1,\ 0]^\top,\ \mathrm{diag}(0.01,\ 10.0)\big) & \text{class } +1, \\ \mathcal{N}\big([-0.1,\ 0]^\top,\ \mathrm{diag}(0.01,\ 10.0)\big) & \text{class } -1. \end{cases} \tag{13}$$

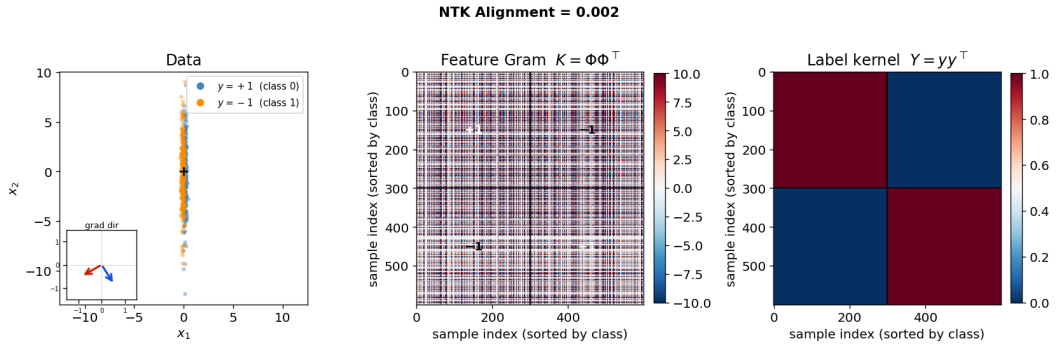

Figure 15: A toy 2D example of low NTK alignment.

The class means now nearly coincide (separated by only 0.2 along x_1), while the variance along x_2 is very large ($= 10$) and shared by both classes. Every pairwise inner product $K_{ij} = \phi_i^\top \phi_j$ is dominated by the x_2 component, which is uninformative about class identity. Consequently, both same-class and different-class pairs produce similar inner products, and the Gram matrix (middle panel of Figure 15) shows no block structure. The inset in the left panel confirms this: because $\bar{\phi}_{+1} \approx \bar{\phi}_{-1}$, the readout gradient $\bar{\phi}_c - \bar{\phi}$ is negligibly small in magnitude. Moreover, although the expected gradient points along x_1 in the limit of many samples, each individual feature vector ϕ_i is dominated by its large x_2 component (variance $= 10.0$). In any finite mini-batch, the x_2 fluctuations therefore swamp the tiny x_1 signal, so the effective gradient direction is not reliably aligned with the label-discriminative axis. The resulting alignment with Y (right panel) is low, even though the two classes are in principle linearly separable along x_1.

A.5.2 Connection between NTK-label alignment and readout learning dynamics

A central challenge in understanding representation learning is bridging *performance-level* quantities (e.g., training loss or classification accuracy) and *representation-level* quantities (e.g., the geometry of the feature space). The former depends on the joint dynamics of the encoder and the readout, as well as on the choice of optimizer, making it difficult to attribute changes in performance to changes in representations alone. The latter, on the other hand, can be computed as a static snapshot at any checkpoint, but does not obviously reflect what the optimizer is doing.

NTK-label alignment offers one principled bridge between these two levels. As we show below, under a set of idealized assumptions, it is *exactly* proportional to the initial rate of loss decay when training a linear readout on top of frozen features. In practice, these assumptions are not fully satisfied,

but alignment continues to serve as a tractable and interpretable proxy for how well the current representations support linear classification.

Setup and notation. Consider an encoder $\phi_\theta : \mathcal{X} \to \mathbb{R}^N$ and a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with C classes. We collect the encoder outputs into the feature matrix $\Phi \in \mathbb{R}^{N\times n}$, whose i-th column is $\phi_i = \phi_\theta(\mathbf{x}_i) \in \mathbb{R}^N$. The linear readout is parameterized by $W \in \mathbb{R}^{C\times N}$, so that the matrix of logits is $W\Phi \in \mathbb{R}^{C\times n}$, with column i giving the logit vector $W\phi_i \in \mathbb{R}^C$ for sample i. We collect the one-hot label vectors into the label matrix $Y \in \mathbb{R}^{C\times n}$, whose i-th column is the one-hot encoding of y_i. Finally, the *feature Gram matrix* is $K = \Phi^\top\Phi \in \mathbb{R}^{n\times n}$, with (i, i')-entry $K_{ii'} = \phi_i^\top \phi_{i'}$.

We make the following assumptions throughout this section:

- **A1. Frozen encoder.** The encoder parameters θ are held fixed; only W is trained.
- **A2. MSE loss.** The training objective is

$$\ell(W) = \frac{1}{2n}\|W\Phi - Y\|_F^2, \tag{14}$$

  where $\|\cdot\|_F$ denotes the Frobenius norm.
- **A3. Gradient flow.** Training follows continuous-time gradient descent, i.e., $\dot{W} = -\frac{\partial \ell}{\partial W}$.
- **A4. Zero initialization.** $W(0) = 0$.
- **A5. Mean-zero features.** $\Phi\mathbf{1} = 0$, i.e., the feature vectors are centered across the dataset.
- **A6. Balanced classes.** $Y\mathbf{1} = 0$, i.e., each class contains the same number of samples.

Assumptions A1–A4 are needed to connect alignment to the loss decay rate. Assumptions A5–A6 additionally ensure that the centered and uncentered versions of the alignment numerator coincide; we discuss this further below.

Gradient flow on the readout. Under A1–A3, the gradient of ℓ with respect to W is

$$\frac{\partial \ell}{\partial W} = \frac{1}{n}(W\Phi - Y)\Phi^\top, \tag{15}$$

so the gradient flow reads

$$\dot{W} = -\frac{1}{n}(W\Phi - Y)\Phi^\top. \tag{16}$$

It is convenient to track the *residual* $R = W\Phi - Y \in \mathbb{R}^{C\times n}$, which measures the discrepancy between the current predictions and the labels. Since the encoder is frozen ($\dot{\Phi} = 0$), differentiating R with respect to time gives $\dot{R} = \dot{W}\Phi$, and substituting the gradient flow equation yields

$$\dot{R} = -\frac{1}{n} R\,K. \tag{17}$$

This is a matrix ODE: the residual is driven by right-multiplication by the Gram matrix K. It decouples across classes (rows of R) but couples samples (columns of R) through K.

Alignment as initial loss decay rate. We now differentiate the loss $\ell(t) = \frac{1}{2n}\|R(t)\|_F^2$ along the gradient flow. Using the residual ODE,

$$\dot{\ell} = \frac{1}{n}\langle R,\, \dot{R}\rangle_F = -\frac{1}{n^2}\operatorname{tr}(R\,K\,R^\top), \tag{18}$$

where $\langle A, B\rangle_F = \operatorname{tr}(A^\top B)$ denotes the Frobenius inner product. Evaluating at $t = 0$ and using A4 ($R(0) = W(0)\Phi - Y = -Y$), we obtain the *initial loss decay rate*:

$$-\dot{\ell}(0) = \frac{1}{n^2}\operatorname{tr}(Y\,K\,Y^\top). \tag{19}$$

We now connect this to the NTK-label alignment defined in Equation 8. Recall that the alignment numerator is $\mathrm{HSIC}(K_{\mathrm{NTK}}, Y)$ (see Section A.1.4 for definitions), which after reduction through the block-diagonal structure of K_{NTK} and Y is proportional to $\operatorname{tr}(HKH \cdot YY^\top H)$, where $H =$

$I - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$ is the centering matrix satisfying $H^2 = H$ and $H^\top = H$. Under A5–A6, the feature vectors and label vectors are already mean-zero, so $HK = K$ and $HY = Y$, and therefore

$$\mathrm{tr}(HKH \cdot YY^\top H) = \mathrm{tr}(KYY^\top) = \mathrm{tr}(Y\,K\,Y^\top), \tag{20}$$

which matches Equation 19 up to the constant $\frac{1}{n^2}$. Hence, under A1–A6:

$$\boxed{-\dot{\ell}(0) \;\propto\; \mathrm{HSIC}(K_{\mathrm{NTK}},\, Y).} \tag{21}$$

The full CKA normalization in Equation 8 divides by $\sqrt{\mathrm{HSIC}(K_{\mathrm{NTK}}, K_{\mathrm{NTK}})\,\mathrm{HSIC}(Y, Y)}$, rendering the alignment invariant to rescaling of Φ (e.g., changes in feature norm across training). This normalization is important for comparability across checkpoints and models, but does not affect the fundamental interpretation.

Interpretation. Under A1–A6, the (last-layer) NTK-label alignment at a given checkpoint answers the following counterfactual question: If the encoder were frozen at this checkpoint and a fresh linear readout were trained from zero with MSE loss under gradient flow, how fast would the loss decay initially? Concretely:

- **High alignment** means the label directions are concentrated in the high-eigenvalue subspace of K. A linear readout trained on these frozen features would make rapid initial progress, indicating that the representations are already well-organized for linear classification.
- **Low alignment** means the label directions are diffuse or concentrated in the low-eigenvalue subspace of K. A linear readout would make slow initial progress, indicating that the encoder has not yet structured its representations in a label-relevant way.

Scope and limitations. The above connection is exact only under A1–A6, which are idealized. In practice:

- **Moving encoder (A1).** During end-to-end training, Φ evolves jointly with W. The alignment at epoch t is therefore a snapshot of an instantaneous quantity, not a description of the full dynamics. Nevertheless, it characterizes how amenable the *current* representations are to linear readout learning.
- **Loss choice (A2).** For cross-entropy loss the residual ODE acquires a nonlinear dependence on the current logits, breaking the exact proportionality. Alignment remains a meaningful proxy but is no longer exactly tied to the loss decay rate.
- **Initialization (A4).** The identification $R(0) = -Y$ relies on $W(0) = 0$. If the readout has already been partially trained (as is the case when measuring alignment mid-training), the connection is approximate.
- **Normalization (A5–A6 and CKA).** The CKA normalization removes scale information. Two checkpoints with equal alignment but different $\|K\|_F$ would yield different raw loss decay rates. Additionally, if features or classes are not perfectly centered or balanced, the centered and uncentered alignment numerators differ by terms involving the mean feature vector and class frequencies.

Despite these caveats, NTK-label alignment has been used as an empirical probe of representation quality throughout the learning dynamics literature [45, 46, 47], and our derivation provides a precise theoretical grounding for its interpretation in the idealized linear-frozen-encoder regime.

B Architecture, Task, and Training Details

We provide full architecture and training details for all experiments in the main text. For the grokking experiments, we consider four tasks: modular addition (Section B.1.1), permutation composition (Section B.1.2), sparse parity (Section B.1.3), and MNIST digit classification (Section B.1.4). Each task is evaluated on an MLP; modular addition, permutation composition, and sparse parity are additionally evaluated on a Transformer, giving seven task-model pairs in total. For the double descent experiment (Section B.2), we use a ResNet-18 trained on CIFAR-10 with varying levels

of label noise. All hyperparameters were chosen to reproduce the grokking and double descent phenomena reported in prior work [2, 3, 7], and are held fixed across seeds unless otherwise noted. Computing resources are described in Section B.3.

B.1 Grokking examples

Throughout this section, β denotes an output scale parameter that multiplies the weights of the final linear layer at initialisation. Concretely, if W_{out} is the last layer's weight matrix, it is initialised as $\beta \cdot \tilde{W}_{\text{out}}$ where $\tilde{W}_{\text{out}}$ follows the standard initialisation scheme. Large β pushes the model into the rich (feature-learning) regime and tends to produce grokking; small β keeps the model in the lazy (kernel) regime and suppresses it. To keep gradient magnitudes comparable across values of β, the optimizer learning rate is rescaled to $\eta = \eta_{\text{cfg}}/\beta$, where η_{cfg} is the value specified in the config.

B.1.1 Modular addition

Task. The task consists of predicting $(a + b) \bmod p$, where pairs (a, b) with $a > b$, $a, b \in \{0, \ldots, p-1\}$ are fed to the model. For the Transformer, an additional constant cue token is appended to the input sequence, following [2, 9]. For both architectures we set $p = 113$, giving a total dataset of $p(p-1)/2 = 6{,}328$ unique pairs.

Architectures. The **MLP** takes the pair (a, b) as input and sums their embeddings before passing them through two linear layers:

$$\mathbf{x} = \mathbf{e}_a + \mathbf{e}_b, \quad \mathbf{h} = \mathrm{ReLU}(W_{\text{in}}\,\mathbf{x}), \quad \mathbf{z} = \mathrm{ReLU}(W_{\text{out}}\,\mathbf{h}), \quad \hat{y} = W_{\text{unemb}}\,\mathbf{z},$$

where $\mathbf{e}_a, \mathbf{e}_b \in \mathbb{R}^{256}$ are learned embeddings ($d_{\text{emb}} = 256$), $W_{\text{in}} \in \mathbb{R}^{128\times 256}$, $W_{\text{out}} \in \mathbb{R}^{256\times 128}$, and $W_{\text{unemb}} \in \mathbb{R}^{p\times 256}$.

The **Transformer** is a one-block decoder-only model with $d_{\text{model}} = 256$, 4 attention heads ($d_{\text{head}} = 64$), and an MLP layer with hidden dimension $d_{\text{mlp}} = 1024$ and ReLU activation. A single LayerNorm is applied to the final representation before the unembedding layer; there is no LayerNorm inside the attention or MLP sub-layers. The input sequence is $[a,\ b,\ \text{sep}]$ with learned positional embeddings ($d_{\text{vocab}} = p + 1 = 114$).

Training. All models are trained with AdamW [48], cross-entropy loss, $(\beta_1, \beta_2) = (0.9, 0.98)$, weight decay 1.0, batch size 200, and a cosine learning rate schedule. Table 1 summarises the per-condition hyperparameters; all other settings are shared.

Table 1: Hyperparameters for the modular addition task.

	MLP		**Transformer**	
	Grokking	Non-grokking	Grokking	Non-grokking
β	1	0.005	2	0.3
lr (config, η_{cfg})	5×10^{-4}	5×10^{-5}	1×10^{-3}	1×10^{-3}
lr (effective, η_{cfg}/β)	5×10^{-4}	1×10^{-2}	5×10^{-4}	3.3×10^{-3}
Epochs	2,000	1,000	500	500
Train samples	3,164 (50%)	3,797 (60%)	3,797 (60%)	3,797 (60%)

B.1.2 Permutation composition

Task. The task consists of predicting the composition $\sigma \circ \tau$ of two permutations drawn from the symmetric group S_5, which has $5! = 120$ elements. The input is the ordered pair (σ, τ), giving a total dataset of $120^2 = 14{,}400$ pairs. For the Transformer, a constant cue token is appended to the input sequence.

Architectures. The **MLP** embeds each permutation separately ($d_{\text{emb}} = 256$), concatenates the two embeddings, and passes the result through two linear layers with tied output weights:

$$\mathbf{x} = [\mathbf{e}_\sigma;\, \mathbf{e}_\tau], \quad \mathbf{h} = \mathrm{ReLU}(W_{\text{in}}\,\mathbf{x}), \quad \mathbf{z} = W_{\text{out}}\,\mathbf{h}, \quad \hat{y} = W_{\text{emb}}^{\top}\mathbf{z},$$

where $\mathbf{x} \in \mathbb{R}^{512}$, $W_{\text{in}} \in \mathbb{R}^{512\times512}$, $W_{\text{out}} \in \mathbb{R}^{256\times512}$, and the output logits are produced via tied embedding weights $W_{\text{emb}}^{\top} \in \mathbb{R}^{120\times256}$.

The **Transformer** shares the same architecture as the modular addition Transformer: a one-block decoder-only model with $d_{\text{model}} = 256$, 4 attention heads ($d_{\text{head}} = 64$), $d_{\text{mlp}} = 1024$, ReLU activation, and a single final LayerNorm. The input sequence is $[\sigma, \tau, \text{sep}]$ with learned positional embeddings ($d_{\text{vocab}} = 121$).

Training. All models are trained with AdamW [48], cross-entropy loss, and $(\beta_1, \beta_2) = (0.9, 0.98)$. The MLP uses a cosine learning rate schedule; the Transformer uses a flat (constant) learning rate. Table 2 summarises the per-condition hyperparameters.

Table 2: Hyperparameters for the permutation composition task.

	MLP		**Transformer**	
	Grokking	Non-grokking	Grokking	Non-grokking
β	1	0.01	1	0.1
lr (config, η_{cfg})	1×10^{-3}	1×10^{-5}	1×10^{-3}	1×10^{-3}
lr (effective, η_{cfg}/β)	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-2}
Weight decay	1.0	1.0	1.0	0.01
Batch size	2,000	2,000	200	200
Scheduler	cosine	cosine	flat	flat
Epochs	10,000	1,000	400	1,000
Train samples	6,000 (42%)	10,800 (75%)	5,760 (40%)	10,800 (75%)

B.1.3 Sparse parity

Task. The sparse parity task is a binary classification problem over binary vectors $\mathbf{x} \in \{\pm1\}^n$ with $n = 40$. The label is the parity of $k = 3$ fixed bits from a randomly chosen support set $S \subset \{1, \ldots, n\}$ with $|S| = k$: $y = \prod_{i\in S} x_i$. The support set is fixed across all examples and is unknown to the model.

Architectures. The **MLP** is a single hidden-layer network with ReLU activation:

$$\hat{y} = W_{\text{out}}\,\text{ReLU}(W_{\text{in}}\,\mathbf{x}),$$

where $W_{\text{in}} \in \mathbb{R}^{1000\times40}$ and $W_{\text{out}} \in \mathbb{R}^{1\times1000}$ (no output bias). The model produces a scalar logit.

The **Transformer** follows the architecture of [8]. It is a single-block encoder with a CLS token at position 0 followed by the 40 bit positions ($n_{\text{ctx}} = 41$, $d_{\text{vocab}} = 3$). The model has $d_{\text{model}} = 1024$, with 128 attention heads each of per-head dimension $d_{\text{attn}} = 8$ and an MLP layer with $d_{\text{mlp}} = 1024$ and GELU activation. There is no LayerNorm. Attention is CLS-only: only the CLS token produces queries, attending over all positions; its updated representation is passed to the MLP and unembedding layers.

Training. The MLP is trained with SGD and hinge loss; the Transformer is trained with AdamW [48] ($(\beta_1, \beta_2) = (0.9, 0.98)$) and cross-entropy loss. The Transformer uses a linear warmup of 10 steps followed by a constant learning rate. Since the MLP and Transformer use different optimizers and loss functions, the full per-condition configuration is listed in Table 3.

B.1.4 MNIST digit classification

Task. The task is 10-class classification of handwritten digit images from the MNIST dataset [49]. Each input is a 28×28 grayscale image (784 features) and the label is one of 10 digit classes. The full MNIST training set contains 60,000 images; both the grokking and non-grokking conditions use a random subset, with the test set subsampled to the same size as the training set.

Architecture. A three-layer MLP with two hidden layers of width 200 and ReLU activations:

$$\hat{y} = W_3\,\text{ReLU}(W_2\,\text{ReLU}(W_1\,\mathbf{x})),$$

Table 3: Hyperparameters for the sparse parity task. Training sample counts have no natural percentage since examples are drawn i.i.d. from $\{\pm 1\}^{40}$.

	MLP		**Transformer**	
	Grokking	Non-grokking	Grokking	Non-grokking
β	1	0.1	1	1
Optimizer	SGD	SGD	AdamW	AdamW
Loss	hinge	hinge	cross-entropy	cross-entropy
lr (config, η_{cfg})	0.1	0.1	1×10^{-3}	5×10^{-4}
lr (effective, η_{cfg}/β)	0.1	1.0	1×10^{-3}	5×10^{-4}
Weight decay	0.01	0.01	0.1	0.01
Batch size	50	50	512	512
Epochs	2,000	2,000	3,000	1,000
Train samples	1,200	5,000	1,000	5,000

where $W_1 \in \mathbb{R}^{200\times 784}$, $W_2 \in \mathbb{R}^{200\times 200}$, $W_3 \in \mathbb{R}^{10\times 200}$. At initialisation all weight matrices are scaled globally by a factor α (the *initialization scale*):

$$W_i \leftarrow \alpha \cdot \tilde{W}_i, \quad \tilde{W}_i \sim \text{standard init.}$$

Unlike the other tasks, the lazy–rich axis here is controlled by α applied to *all* layers, not by the output-layer β parameter. Accordingly $\beta = 1$ for both conditions and the lr-rescaling $\eta = \eta_{\text{cfg}}/\beta$ does not apply.

Training. All models are trained with AdamW [48], MSE loss on one-hot targets, $(\beta_1, \beta_2) = (0.9, 0.999)$, learning rate 1×10^{-3}, weight decay 2×10^{-4}, batch size 200, and a constant learning rate schedule (no scheduler). Table 4 summarises the per-condition hyperparameters.

Table 4: Hyperparameters for the MNIST digit classification task.

	Grokking	Non-grokking
Initialization scale (α)	6.0	1.0
Epochs	200,000	1,000
Train samples	1,000 (1.7%)	10,000 (16.7%)

B.2 Double descent example

Task. The task is 10-class image classification on CIFAR-10 [50], which contains 50,000 training images and 10,000 test images of size $32 \times 32 \times 3$. We reproduce the epoch-wise double descent phenomenon of Nakkiran et al. [7]: test error first decreases, then increases, and finally decreases again as training continues. To induce this behavior, a fraction of training labels are corrupted: each noisy example is assigned a uniformly random incorrect class, with the corruption fixed at the start of training (the same wrong label is used across all augmented views). We consider three noise levels: 0%, 10%, and 20%.

Architecture. We use a ResNet18 adapted for CIFAR-10 spatial resolution. The standard conv1 layer (7×7, stride 2) is replaced by a 3×3, stride-1 convolution, and the max-pool layer is replaced by an identity map, so the feature map resolution is preserved at 32×32 through the first residual group. The spatial flow is 32→32→16→8→4→1 (with average pooling), yielding a 512-dimensional pre-classifier feature vector followed by a linear output head.

Training. Models are trained with Adam [51] ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$), cross-entropy loss, learning rate 10^{-4} (fixed, no schedule), no weight decay, and batch size 128. Training runs for 4,000 epochs. Data augmentation consists of random crops (padding 4) and random horizontal flips; no augmentation is applied at test time. Checkpoints are saved at 200 log-uniformly spaced epochs.

Table 5: Hyperparameters for the double descent task.

	ResNet18
Optimizer	Adam
(β_1, β_2)	$(0.9,\ 0.999)$
Learning rate	1×10^{-4} (fixed)
Weight decay	0
Batch size	128
Epochs	$4{,}000$
Noise levels	$0\%,\ 10\%,\ 20\%$
Train samples	$50{,}000$

B.3 Computer resources

All model training was run on a single NVIDIA A100 GPU. For small models trained over a few hundred epochs (e.g., ModAdd Transformer at 500 epochs, PermComp Transformer at 400 epochs), each individual training run takes approximately 1–5 minutes. For longer runs (e.g., PermComp MLP grokking at 10,000 epochs, MNIST grokking at 200,000 epochs), training takes roughly 30–60 minutes per run. With 3 seeds × 2 conditions (grokking/non-grokking) across 6 task×architecture pairs, total training compute is on the order of 10–20 GPU-hours. Post-training analysis (metric computation, GLUE manifold analysis, linear probes) runs entirely on AMD EPYC CPU. GLUE computation takes approximately 1 minute per checkpoint per run; across about 200 log-uniformly spaced checkpoints per run, this amounts to roughly 3 CPU-hours per run, parallelized across up to 3,000 CPU cores. Total analysis compute is on the order of 100–200 CPU-hours.

C Further Discussion of Previous Work

We provide further discussion of how our framework relates to four broad threads of prior work on grokking, as well as background on GLUE theory and its previous applications (Section C.5). We address the lazy-rich hypothesis (Section C.1), which frames grokking as a transition between kernel regression and feature learning regimes. We discuss task-specific progress measures (Section C.2), which construct mechanistically interpretable grokking indicators, and contrast their approach with our task-agnostic framework. We compare our measures against other task-agnostic progress measures proposed in the literature (Section C.3), highlighting key differences in interpretability, architectural generality, and empirical robustness. Finally, we review weight-based theories of grokking (Section C.4), discussing both the empirical counterexamples that challenge them and their theoretical limitations as a lens for understanding generalization.

C.1 Relation to lazy-rich hypothesis

The lazy-rich hypothesis by Kumar et al. [10] framed grokking as a transition from kernel regression dynamics with a fixed Neural Tangent Kernel (NTK) to genuine feature learning, in which the kernel itself evolves to better align with the task. Two prior results of note to our work relate to this framing. Firstly, Zheng et al. [13] reported evidence that conflicts with the network behaving as kernel regression with K_{NTK} fixed at initialization before grokking, observing that the kernel distance between the NTK at different epochs, defined for kernel matrices K_1, K_2 as $1 - \frac{\langle K_1, K_2 \rangle_F}{\|K_1\|_F \|K_2\|_F}$, varies meaningfully prior to grokking onset. Secondly, Kumar et al. [10] reported a monotone increase in NTK-label alignment over training in their toy setting of polynomial regression.

Our findings engage with both of these results. Consistent with Zheng et al. [13], we find that both representation learning and readout calibration are actively progressing before grokking onset, hence the learning dynamics do not resemble lazy learning in our setting, providing additional evidence against the fixed-kernel picture (i.e., lazy learning) of pre-grokking training. However, we find a more nuanced picture of the NTK-alignment dynamics reported by Kumar et al. [10]: across our experiments, we observe both monotonically increasing NTK-alignment as well as non-monotonic behavior of NTK-alignment over training (e.g., Figure 18, Figure 21).

C.2 Relation to task-specific grokking analyses

A complementary line of work has developed task-specific progress measures for grokking by exploiting the algebraic structure of particular tasks. Barak et al. [8] introduced the notion of progress measures in the grokking context—scalar quantities that vary more smoothly over training than loss or accuracy, and thus provide earlier and more continuous signal about a network's progress toward generalization. They instantiated this idea for the sparse parity task using a Fourier gap measure derived from the task's Boolean structure. Subsequently, Nanda et al. [9] and Chughtai et al. [19] identified analogous progress measures for modular addition and group composition respectively, in each case leveraging the specific algebraic structure of the task to construct interpretable, mechanistically grounded measures.

Our metrics perform well in revealing the hidden progress of the network across all of these settings without requiring any task-specific analysis or structural assumptions. A point of nuance is that our measures are less directly task-interpretable than their metrics, which admit clean mechanistic readings tied to the learned algorithm. We view this as a natural cost of universality: by not assuming any task structure, our metrics sacrifice some interpretability in exchange for broad applicability and ease of implementation, making them better suited to practical settings where task-specific decompositions are unavailable or prohibitively expensive to derive.

C.3 Relation to other task-agnostic progress measures

Several recent works have proposed task-agnostic progress measures for grokking that, like ours, do not rely on task-specific structural assumptions. Humayun et al. [4] introduced *local complexity*, a measure of the density of linear regions in a network's input-space partition, and showed that it consistently exhibits a descent-ascent-descent pattern across grokking examples. While task-agnostic in principle, the measure is sensitive to hyperparameter choices that must be tuned per architecture and is restricted to networks with continuous piecewise linear activation functions [4, Appendix C]. Merrill & Tsilivis et al. [21] proposed weight sparsity as a progress measure, observing that the sparsification of weights contributing to the network output co-occurs with the onset of grokking. Clauw et al. [52] applied O-Information—an entropy-based measure of higher-order statistical dependencies—to model activations, identifying training phases that correspond to the qualitative behavior of the train and test loss curves during grokking.

Our framework differs from these approaches primarily in interpretability. While the measures described above are effective at empirically detecting grokking, none situates its findings within a broader theoretical framework that explains why generalization is delayed. For instance, Merrill & Tsilivis et al. [21] do not ground weight sparsity in underlying principles connecting it to the network's learning dynamics. By contrast, our measures are grounded in the geometry of learned representations and connect directly to feature learning theory [11], providing an interpretable account of the representational changes that underlie the transition to generalization.

C.4 Relation to weight-based theories of grokking

Weight-based analyses are a natural starting point for understanding neural network behavior, since the weights fully parameterize the learned function [53, 54, 55]. Within the grokking literature, two influential weight-based accounts have been proposed. Liu et al. [56] argued that grokking arises from the dynamics of weights under high initial weight norm: weight decay slowly pushes the network toward a regime of smaller weight norm, with delayed generalization explained by the fact that large weight norm enables overfitting while reducing the marginal penalty of training loss, slowing the transition to generalizing solutions. Varma & Shah et al. [22] offered a complementary mechanistic account, framing grokking as a competition between two circuits with equivalent training loss: a memorization circuit and a slower-to-learn generalizing circuit that achieves lower test loss with smaller weight norm. Under their framework, weight decay creates pressure that eventually favors the more parameter-efficient generalizing circuit, producing the characteristic delay.

More recent work has identified counterexamples to weight-based accounts of grokking. Kumar et al. [10] demonstrate grokking on modular addition trained without weight decay, observing weight norm *increasing* during the transition to generalization, and Golechha et al. [25] reported similar weight norm dynamics for MNIST under a modified loss function. These results suggest that weight norm reduction is neither necessary nor universally diagnostic of grokking.

Beyond this empirical evidence against weight-norm-centric explanations, weight-based theories are by construction blind to the structure of the network's learned representations—the geometric organization of features that ultimately determines what the network has learned. Our framework directly probes this representational structure, providing a complementary lens that captures aspects of grokking dynamics that weight-based analyses cannot.

A further distinction between our framework and existing weight-based accounts of grokking lies in their respective theoretical foundations. While seminal work on generalization [53, 54, 55] provides rigorous complexity bounds, these often involve nuanced spectral or margin-based properties that extend beyond the simple weight-norm heuristics prioritized in current grokking literature [56, 22]. Furthermore, because scalar weight changes do not necessarily correspond to functional changes—for instance, under simple reparameterizations—a more mathematically principled treatment of network dynamics is required. Our approach addresses this by utilizing representational measures with strong theoretical backing in Neural Tangent Kernel (NTK) theory [26] and GLUE theory (see Section C.5), offering a robust lens that captures aspects of the transition to generalization that weight-based analyses may overlook.

C.5 Previous applications of GLUE framework

The Geometry Linked to Untangling Efficiency (GLUE) theory [14], which serves as an extension of earlier work that developed Manifold Capacity Theory [15, 35, 39, 14, 57], provides a suite of geometric measures for characterizing the structure of learned representations, and has been applied successfully to two problem settings closely related to ours. Chou & Le et al. [11] showed that the critical dimension — which is inversely proportional to the capacity metric they use — tracks the network's feature learning, and that GLUE geometry measures more broadly are effective at classifying subtypes of feature learning dynamics. Chou et al. [37] extended this line of work to out-of-distribution generalization, demonstrating that a collection of GLUE measures computed on in-distribution features from the model robustly predicts model performance on OOD data, outperforming a broad set of alternative measures in their comparison. Beyond ML applications, manifold capacity theory has been applied to study representations in biological neural networks, including the macaque visual cortex [58], social learning [41], and olfactory processing circuits [59], demonstrating its utility as a general framework for characterizing representational structure across both artificial and biological systems.

These results provide empirical motivation for our use of critical dimension and GLUE geometry measures to probe the structure of learned representations during grokking. In particular, the demonstrated ability of these measures to discriminate between feature learning regimes and to predict generalization in OOD settings suggests they are well-suited to capturing the representational changes that underlie the delayed generalization characteristic of grokking.

D Additional Results on Diagnostic Measures

We provide additional results for all six task-model pairs in both grokking (Section D.1) and non-grokking (Section D.2) settings across three seeds each. Definitions for the shaded regions used throughout the main text are given in Section D.4.

D.1 Grokking examples

We present grokking examples for all six task-model pairs across three seeds each (Figure 16–Figure 21). For each run, we plot train and test accuracy alongside the critical dimension N_{crit} and the four-phase shading defined in Section D.4. Across all settings, N_{crit} decreases gradually throughout training, consistent with the continuous representation learning reported in Section 3.1. One exception is the SparseParity Transformer, where grokking is observed in only one of three seeds (seed=1); the remaining seeds do not exhibit a clear grokking pattern and are included for completeness.

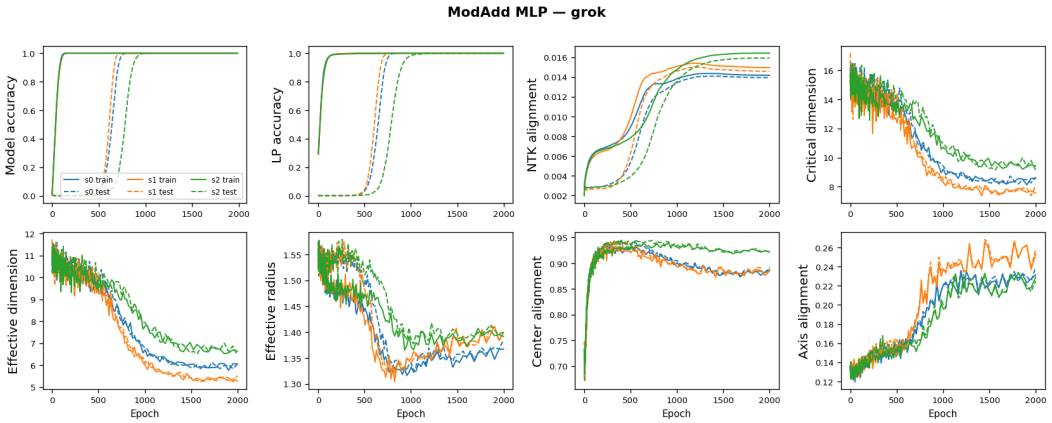

Figure 16: Grokking examples of ModAdd MLP.

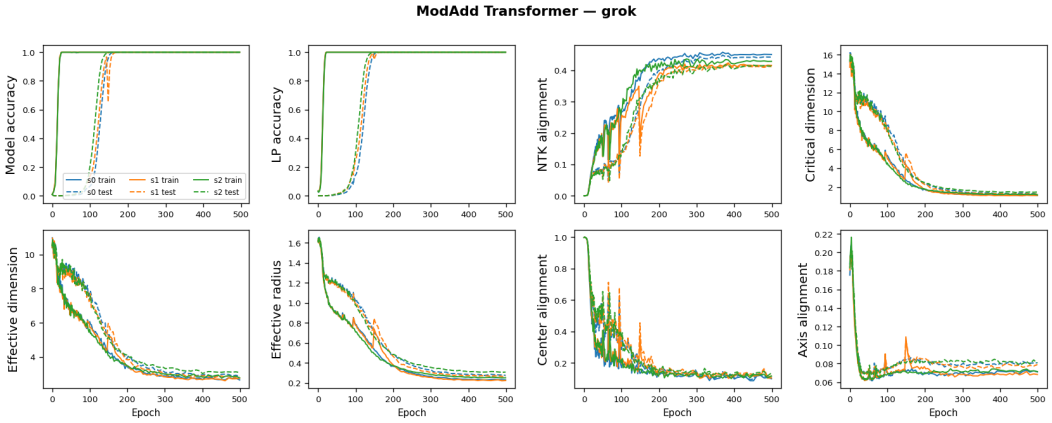

Figure 17: Grokking examples of ModAdd Transformer.

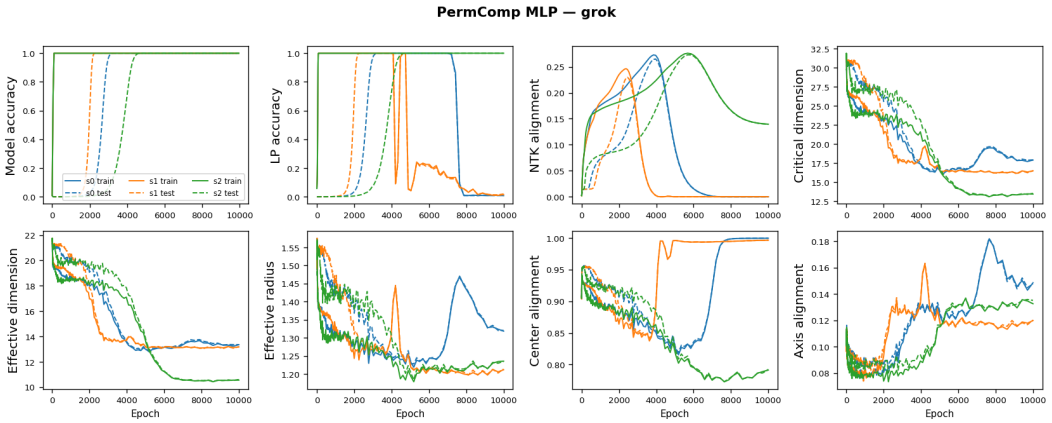

Figure 18: Grokking examples of PermComp MLP.

D.2 Non-grokking examples

We present non-grokking counterparts for all six task-model pairs across three seeds each (Figure 22–Figure 27). These are obtained by modifying the training recipe to eliminate the grokking pattern (see Section D.4 for details).

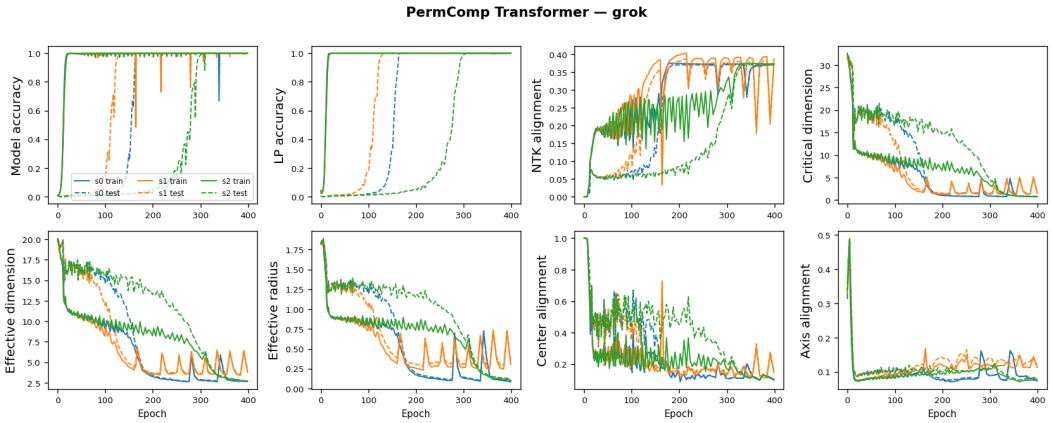

Figure 19: Grokking examples of PermComp Transformer.

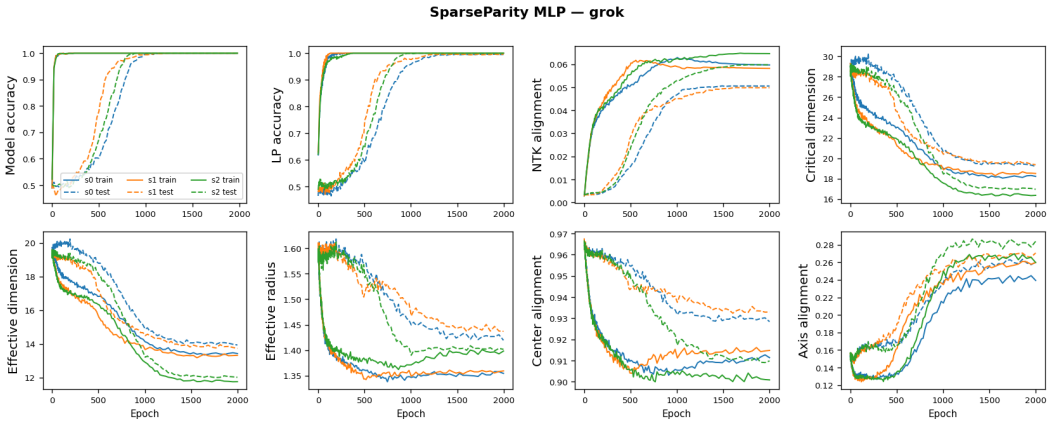

Figure 20: Grokking examples of SparseParity MLP.

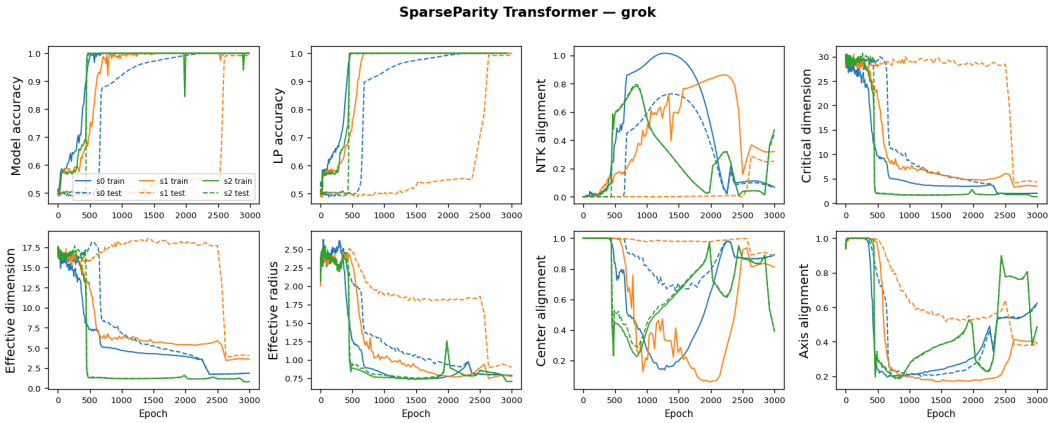

Figure 21: Grokking examples of SparseParity Transformer.

D.3 Epoch-wise double descent through label noises

We present additional results for the epoch-wise double descent experiment across three levels of label noise: 0% (Figure 28), 10% (Figure 29), and 20% (Figure 30). At 0% noise, the double descent pattern and diagnostic signatures are absent, serving as a clean baseline. As label noise increases, the double descent dip in test accuracy becomes more pronounced, and the diagnostic signatures—plateauing

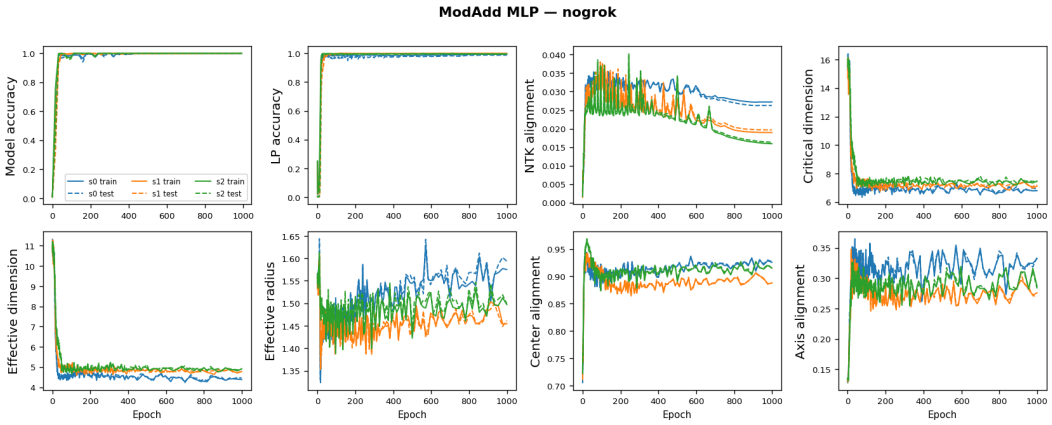

Figure 22: Non-grokking examples of ModAdd MLP.

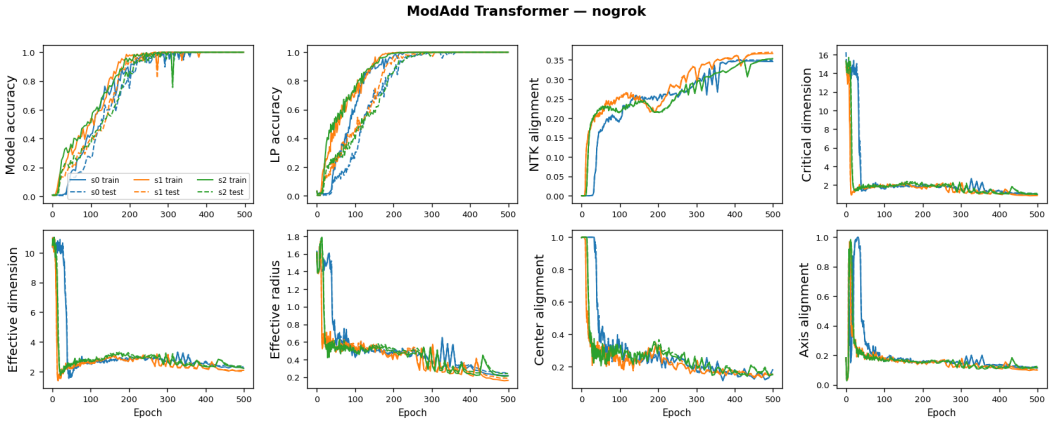

Figure 23: Non-grokking examples of ModAdd Transformer.

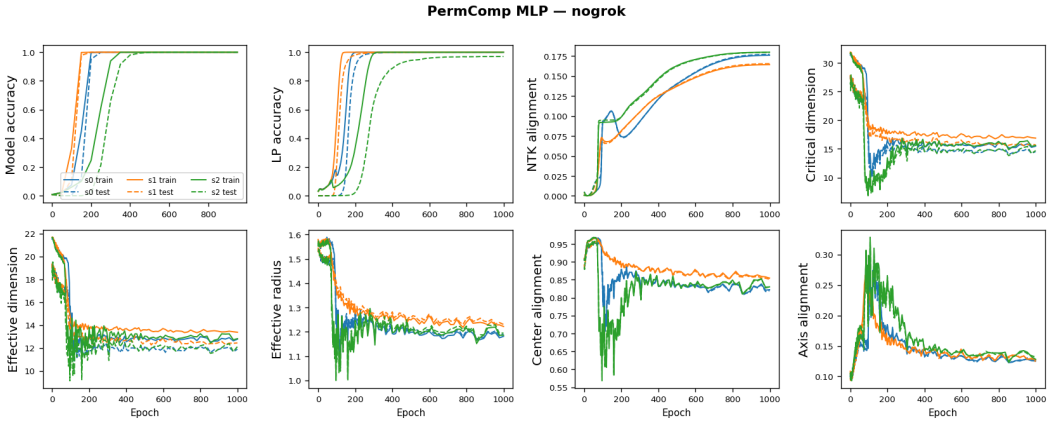

Figure 24: Non-grokking examples of PermComp MLP.

LP accuracy, rising N_{crit}, and increasing train-test NTK alignment gap—become increasingly visible, consistent with the two-stage mechanism described in Section 4.2.

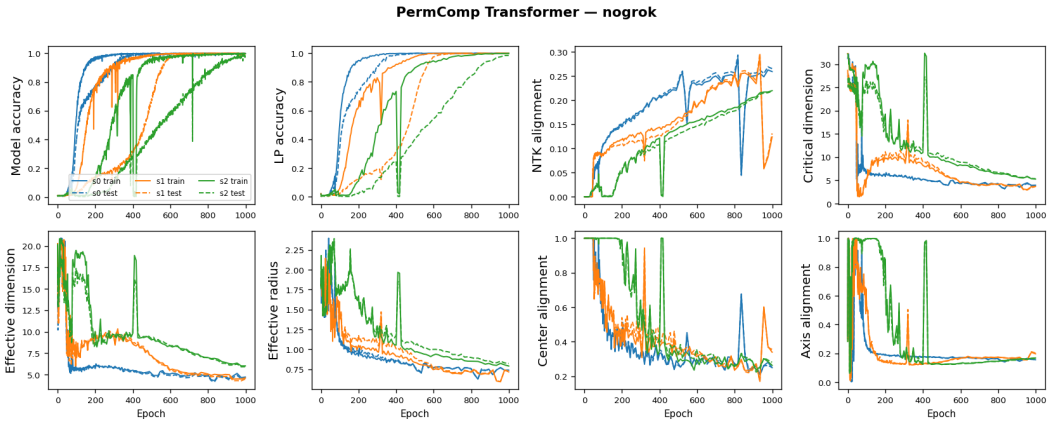

Figure 25: Non-grokking examples of PermComp Transformer.

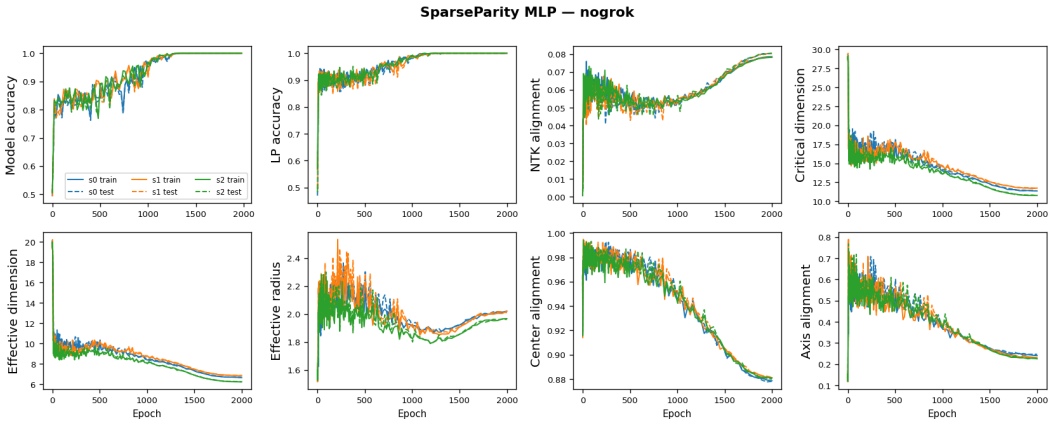

Figure 26: Non-grokking examples of SparseParity MLP.

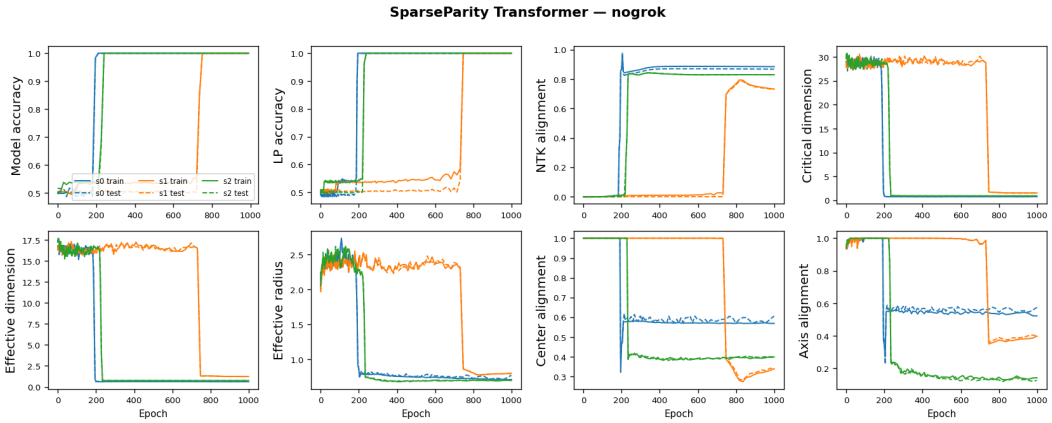

Figure 27: Non-grokking examples of SparseParity Transformer.

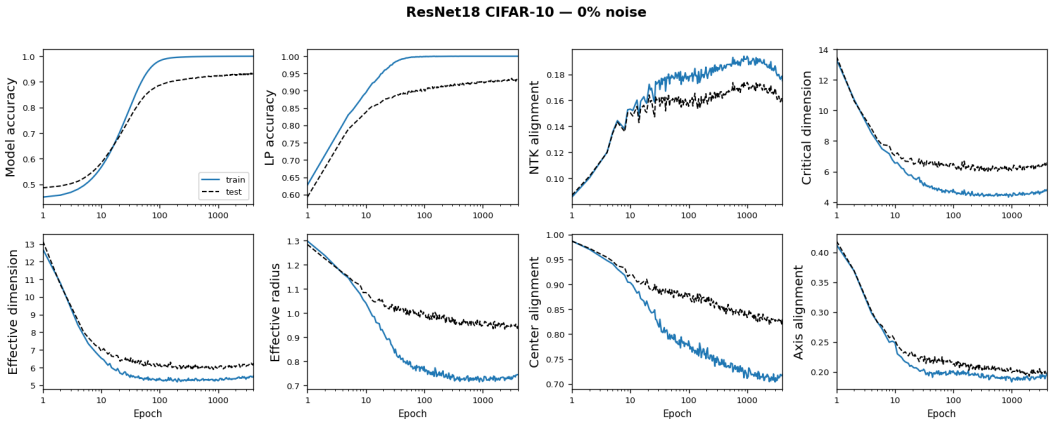

Figure 28: Epoch-wise double descent through label noises, 0% label noise.

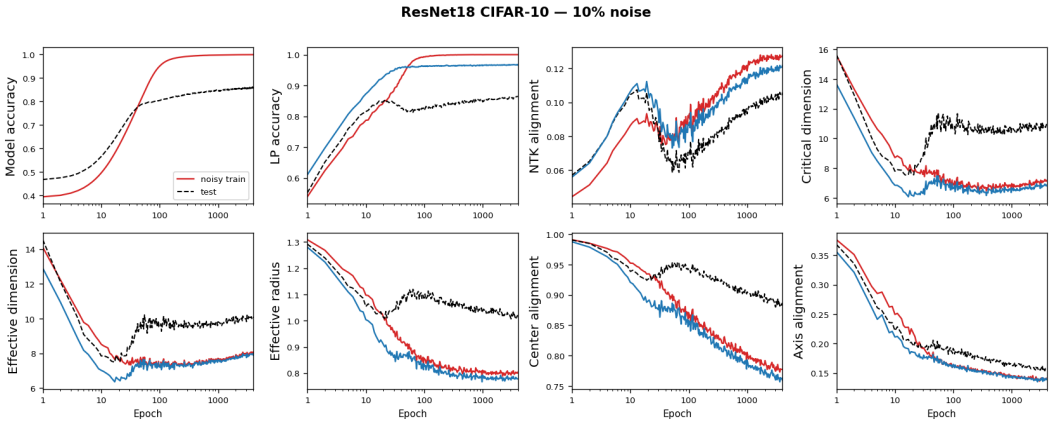

Figure 29: Epoch-wise double descent through label noises, 10% label noise.

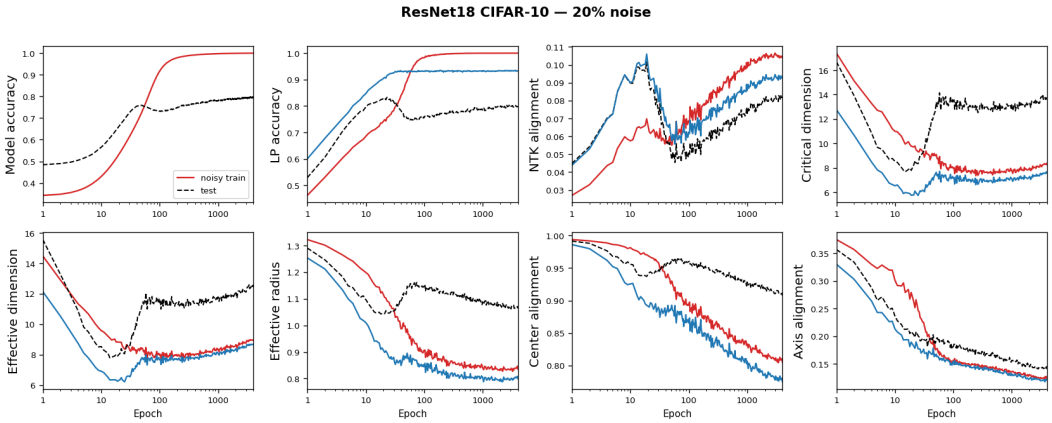

Figure 30: Epoch-wise double descent through label noises, 20% label noise.

D.4 Event detection in the plots in main text

Grokking cases. We identify three critical epochs from the training curves to partition training into four phases. Train-100 is the first epoch at which training accuracy reaches 99% and remains there for at least 3 consecutive checkpoints. Grok onset is the first epoch at which test accuracy exceeds 5% of its eventual maximum (computed on a smoothed curve) and remains above that threshold for at least 3 consecutive checkpoints; for SparseParity, where test accuracy starts near chance, we instead use an absolute threshold of 60%. Grok offset is defined analogously, using 95% of the eventual maximum test accuracy. Smoothing is performed with a uniform moving average of window size 5 prior to threshold crossing detection. These three events divide training into four phases, shown as background shading in all epoch-axis plots: memorization (start $\rightarrow$ train-100, green), post-memorization (train-100 $\rightarrow$ onset, blue), grokking transition (onset $\rightarrow$ offset, orange), and post-grokking (offset $\rightarrow$ end, red).

Non-grokking cases. For models that do not exhibit grokking, we define two critical epochs: train-100, the first epoch at which training accuracy reaches 99% and remains there for at least 3 consecutive checkpoints, and test-100, defined analogously for test accuracy. These two events divide training into three phases, shown as background shading in non-grokking plots: pre-memorization (start $\rightarrow$ train-100, blue), generalization gap (train-100 $\rightarrow$ test-100, yellow), and post-generalization (test-100 $\rightarrow$ end, red). For MNIST, where test accuracy plateaus below 100%, the test-100 threshold is never reached; we therefore use only the train-100 boundary, yielding two phases: memorization (start $\rightarrow$ train-100, green) and post-memorization (train-100 $\rightarrow$ end, blue).

Double descent cases. For double descent experiments (ResNet18 trained on CIFAR-10 with label noise), we identify two critical epochs from the validation accuracy curve. Peak is the epoch of maximum validation accuracy, corresponding to the early-stopping optimum before double descent degrades generalization. Recovery is the first epoch after the subsequent valley at which the smoothed validation accuracy begins a sustained monotone increase, detected as the first point in the post-valley curve where accuracy increases for at least 5 consecutive checkpoints (smoothing window of 15). These two events divide training into three phases: pre-peak (start $\rightarrow$ peak, green), double descent valley (peak $\rightarrow$ recovery, blue), and post-recovery (recovery $\rightarrow$ end, red). Note that for the 0% noise control, where no double descent occurs, we instead use only train-100 as a single dividing event between the memorization and post-memorization phases.

No double descent cases. For the clean training case (0% label noise), no double descent occurs and the validation accuracy increases monotonically after memorization. We therefore use a simpler two-phase scheme: train-100, the first epoch at which training accuracy reaches 99% on the clean labels, divides training into a memorization phase (start $\rightarrow$ train-100, green) and a generalization phase (train-100 $\rightarrow$ end, red). Since there is no valley or recovery event, the double descent phase boundaries do not apply.

E Task-specific analysis

The representation-readout decomposition described in the main text is task-agnostic. In this section, we make a preliminary exploration of how to leverage more fine-grained task structures in the analysis, e.g., the algebraic structure relating different class pairs (for instance, the fact that the label for (a, b) and the label for $(a + 1, b)$ differ by exactly one in $\mathbb{Z}_p$). We take a first step toward incorporating task structure into the analysis. Specifically, we compute the pairwise critical dimension $N_{\text{crit}}(c, c')$ for every label pair (c, c'), obtaining a $C \times C$ matrix that captures the fine-grained geometry of the representation. We then ask whether these structured representations support transfer to a related but distinct task via a frozen linear probe.

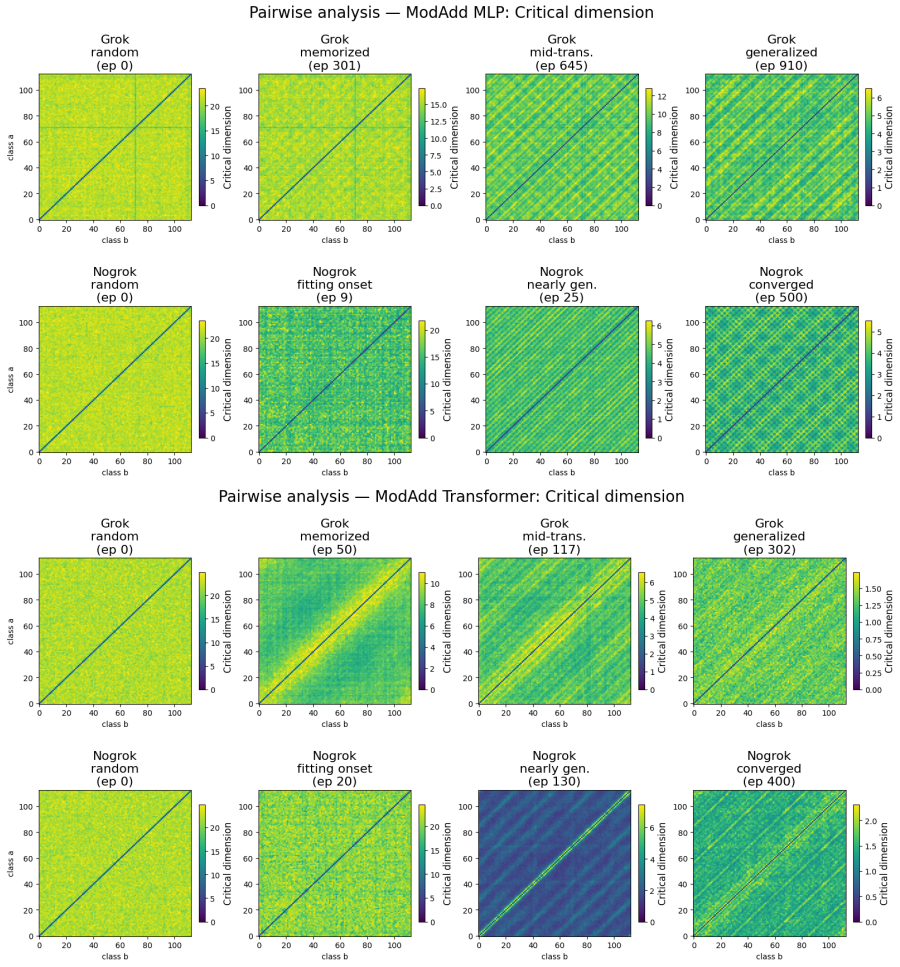

Figure 31: Pairwise critical dimension (N_{crit}) matrices for the ModAdd task. Top: MLP. Bottom: Transformer. Each panel shows a 113×113 matrix (class a vs. class b). Rows correspond to grokking (top) and non-grokking (bottom) conditions; columns correspond to four representative epochs. Colorbars are scaled to the 99th percentile of each panel independently.

E.1 Pairwise critical dimension heatmaps

Figure 31 and Figure 32 show the $p \times p$ critical dimension matrices at four representative epochs for the grokking and non-grokking conditions, for both the MLP and Transformer architectures. For **modular addition** ($p = 113$, Figure 31), the matrices are unordered (rows/columns indexed by residue class $0, \ldots, 112$). For **permutation composition** ($p = 120$, Figure 32), classes are reordered by conjugacy class (cycle type) of the corresponding element of S_5: identity, transpositions, double transpositions, 3-cycles, 4-cycles, (3+2)-cycles, and 5-cycles.

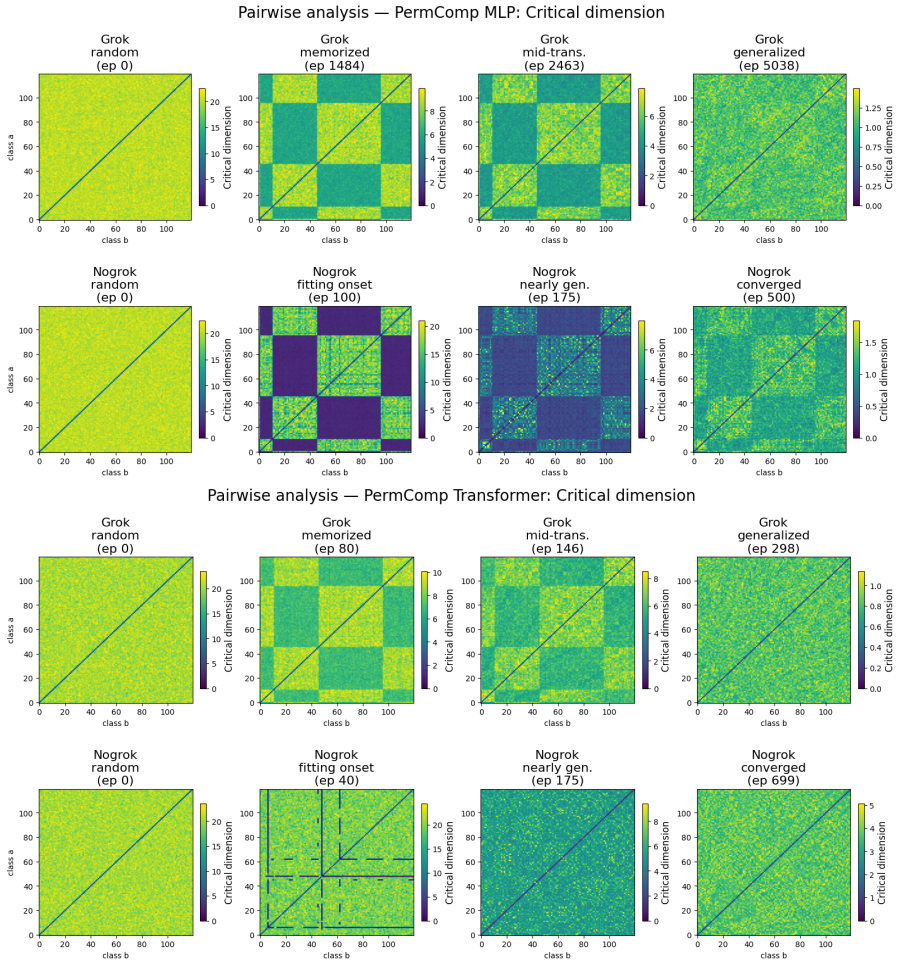

Figure 32: Pairwise critical dimension matrices for the PermComp task. Top: MLP. Bottom: Transformer. Each panel shows a 120×120 matrix with elements of S_5 reordered by conjugacy class (cycle type): identity, transpositions, double transpositions, 3-cycles, 4-cycles, (3+2)-cycles, 5-cycles.

At early epochs, both grokking and non-grokking models show roughly uniform matrices. As training progresses, the grokking models develop visually structured patterns in the pairwise critical dimension matrices. The non-grokking models also develop structure, but the patterns differ qualitatively from those of the grokking models. We do not offer an interpretation of these patterns here; we simply note their existence as a starting point for future investigation.

E.2 Transfer to related tasks

To probe whether the pairwise structure observed above has behavioral consequences, we freeze each trained encoder and fit a linear probe on a related transfer task. For modular addition we predict the *difference* $(a - b) \bmod p$ from all $\binom{p}{2} = 6328$ ordered pairs (a, b) with $a > b$, using an independent 60/40 train/test split (seed 42). For permutation composition, we predict the *inverse composition* $\sigma_i \circ \sigma_j^{-1} \in S_5$ (120 classes) from all 14400 ordered pairs, again with an independent 60/40 split. Both probes are trained with AdamW (lr $= 10^{-3}$, wd $= 1.0$, $\beta = (0.9, 0.98)$, 200 epochs, full batch).

Figure 33 shows the linear probe accuracy on the transfer tasks at the representative epochs. For both tasks and both architectures, the grokked models consistently achieve higher transfer test accuracy than the non-grokking models at late epochs. The two architectures show different profiles: for ModAdd, the MLP grokked model achieves substantially higher transfer accuracy than the Transformer; for

PermComp, the differences are more modest. The mechanism behind these observations is left for future work.

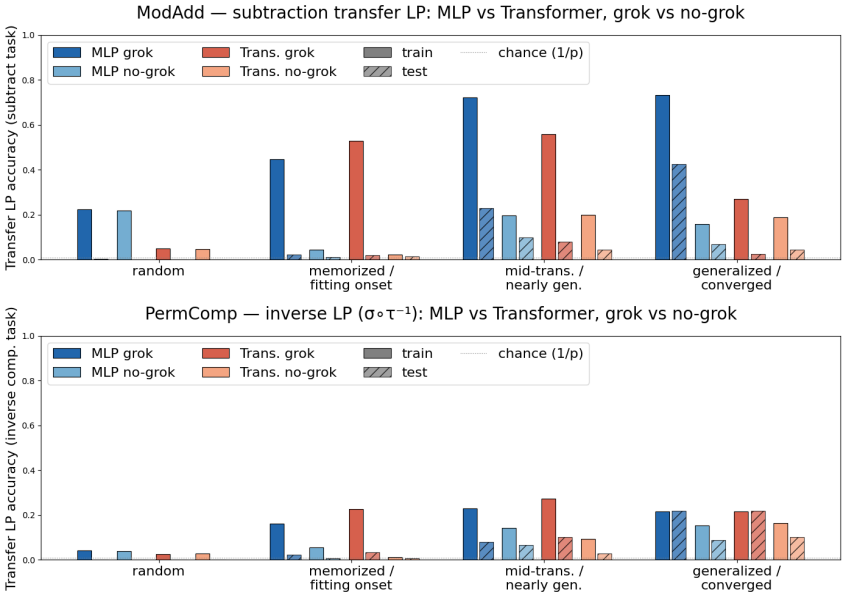

Figure 33: Transfer linear probe accuracy. Top: ModAdd subtraction task ($(a - b) \bmod p$, chance $= 1/113$). Bottom: PermComp inverse composition task ($\sigma \circ \tau^{-1}$, chance $= 1/120$). Solid bars: train accuracy; hatched bars: test accuracy. Each cluster of bars corresponds to one training stage.

E.3 Discussion

The pairwise critical dimension matrices reveal that grokked and non-grokked representations organize class-pair geometry differently. While the mechanisms behind these patterns are unclear, we observe functional consequences via transfer linear probes (Figure 33): grokked models consistently outperform non-grokking ones at late training stages. These findings are preliminary and descriptive; we view them as a starting point for a more principled, task-aware representation analysis.

F Metric Consistency Quantification

In our analysis framework, we use several representation-based measures (e.g., NTK alignment, critical dimension, GLUE geometric measures) to probe the learning dynamics. These metrics indicate that learning progresses continuously throughout training, rather than exhibiting the sharp phase transition suggested in prior work using the lazy-to-rich account [10]. They also provide a stable signal for tracking learning dynamics over time [11]. In particular, even in the presence of grokking—characterized by a delayed and rapid increase in test accuracy relative to training accuracy— the train and test curve of these representation metrics closely mirrors each other during training. This suggests that representation metrics offer a more consistent and informative measure of training progress than standard loss or accuracy curves.

F.1 Implementation and definition of the consistency measure

We quantify consistency between train and test curves by measuring their co-movement via correlation [60]. To reduce noise, we first smooth each trajectory using a cubic smoothing spline, which provides a stable, nonparametric estimate of the underlying trend by balancing data fidelity and curvature [61, 62]. Formally, the smoothing spline is defined as the solution to the regularized least-squares problem:

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int (f''(u))^2 \, du,$$

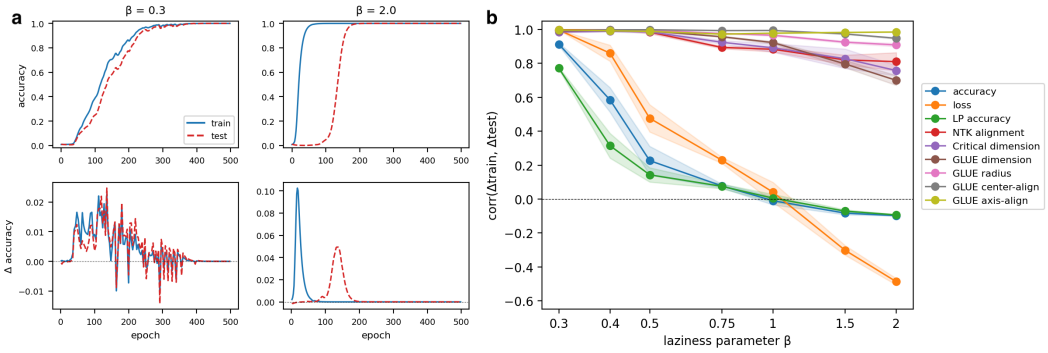

Figure 34: Consistency between train and test curves across different values of the laziness parameter β. Consistency is defined as the correlation between the first differences of the train and test curves after spline smoothing. **a**, Time series of train and test curves for different values of β, along with their corresponding first differences. **b**, For standard metrics (loss and accuracy), this correlation decreases and eventually becomes negative as β increases, indicating the onset of grokking. In contrast, NTK alignment, critical dimension, and GLUE geometric measures maintain consistently high correlation, providing a more reliable signal for tracking learning dynamics. Shaded regions denote the standard deviation across random seeds.

where f is a spline function and λ is a regularization parameter selected via generalized cross-validation. We then compute first differences of the smoothed trajectories, $\Delta X_t = X_t - X_{t-1}$, and measure consistency as the Pearson correlation between train and test dynamics:

$$\mathrm{Corr}\left(\Delta X_t^{\text{train}},\ \Delta X_t^{\text{test}}\right),$$

where X_t denotes the smoothed time series of the metric of interest (e.g., loss, accuracy, LP accuracy, or GLUE score) over training.

F.2 Consistency of different metrics across learning rates

In Figure 34, we report the consistency measure across different learning rates for the Modular Addition task using a Transformer architecture. Following Kumar et al. [10], the laziness parameter β (denoted α in their work) controls the extent of grokking, with larger values inducing stronger effects. Specifically, β is implemented by rescaling the output logits, i.e., multiplying the network outputs by a factor β and dividing the learning rate by a factor of β, thereby inducing lazy training for large β.

As shown in Figure 34, for standard metrics such as loss and accuracy, the correlation between train and test dynamics becomes negative when $\beta > 1$, indicating the onset of grokking. In contrast, for the representation metrics considered in our analysis framework (NTK alignment, critical dimension, GLUE geometric measures), the correlation remains consistently high and close to 1 across all learning rates. These results suggest that the proposed consistency measure captures the grokking phenomenon, while GLUE representation metrics provide a stable and reliable signal for tracking learning dynamics over time.

F.3 Consistency across tasks, architectures, and training stages

In Figure 35, we report the consistency measure across different tasks and architectures. For standard metrics such as loss and accuracy, the correlation between train and test dynamics is often negative or close to zero, indicating the presence of grokking and poor alignment of the train and test curve in learning dynamics. In contrast, the representation metrics described above exhibit consistently positive high correlations across most tasks and architectures. The last two rows of Figure 35 further analyze consistency across different stages of training. We observe that the onset of grokking coincides with a drawdown of the correlation between train and test dynamics.

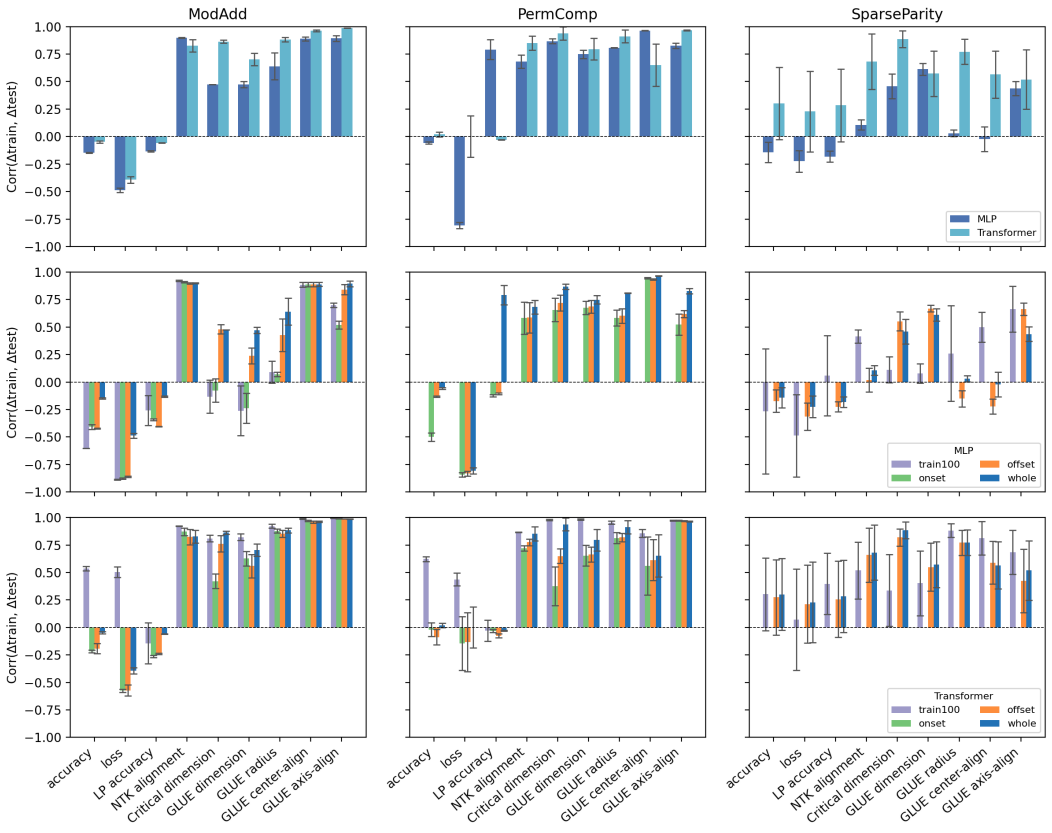

Figure 35: Consistency between the test and train curve across tasks, architectures, and training stages. The top row reports overall consistency across different tasks and architectures. Standard metrics (loss and accuracy) exhibit predominantly negative correlations, while representation metrics (NTK alignment, critical dimension, and GLUE geometric measures) maintain high consistency, close to 1. The middle and bottom rows further analyze consistency computed over different training segments (train100, onset, offset, and full trajectory). The onset and offset phases of grokking significantly lowers the consistency. Error bars indicate the standard error of the mean across seeds.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main results are presented and justified in Section 3 and Section 4. More details are provided in Appendix A and Appendix D.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of this work in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: All details (definitions, derivations, examples) of the theoretical results are provided in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all experimental details for reproducibility in Appendix B. We will provide a GitHub repo along with the camera-ready version of the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will provide a GitHub repo that gives open access to data and code along with the camera-ready version of the paper.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental details are provided in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide seed-average results in the main text and show all the seed individual results in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Details for computer resources are provided in Section B.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We confirm that the research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the paper's broader impact in Section 5.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cited, to the best of our knowledge, all papers that we used/mentioned. Appendix C contains detailed discussions and citations on related work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: N/A

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.