STRUCTURE-ALIGNED STEERING: TASK-SPACE OPERATIONS FOR ACTIVATION CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Activation steering usually linearly adds a vector to a hidden state, imposing translation that disregards the inherent structure of the task (e.g., cyclical structure of weekdays). We introduce STRUCTURE-ALIGNED STEERING, an encoder-decoder interface that maps hidden states into task-structured coordinates, applies transformations there, and returns model-readable states. With prescribed weekday coordinates, one rotation advances the output toward the next day, composes fractional steps with smooth output transfer between neighboring days, and closes the cycle, which linear steering cannot achieve. Without clear coordinate targets, matching geometry-derived output distributions still yields compatible geometry: sparse number mixtures unfold into a continuous circle, and color encodings preserve RGB-cube neighborhoods and approximate distances, which the decoder carries into hidden states. Perturbing these coordinates or supervising with task-inconsistent geometry reduces accuracy, linking this organization to steering. To show practical value beyond mechanistic analysis, we build a steerer–aligner system for modular arithmetic that outperforms parameter-matched PEFT baselines across eight models. Task-space alignment thus makes steering structured and compositional rather than merely directional.

1 INTRODUCTION

Activation steering controls a parameter-frozen language model by adding a concept direction to a hidden state (Turner et al., 2023; Zou et al., 2023; Panickssery et al., 2023). Evidence for linear concept representations supports estimating this direction (Marks & Tegmark, 2024; Park et al., 2023), but does not imply that $h \mapsto h + v$ is the conceptually correct operation. For example, as shown in Figure 1A, in the weekday scenario, repeated nonzero additions compose by translation and never return to h, whereas a one-day advance must wrap from Sunday to Monday and close after seven steps. The mismatch is also geometric: distances and mixtures of hidden states or unembeddings need not reproduce semantic relations, so a red–blue midpoint need not yield purple.

Endpoint evaluations can conceal this mismatch: a correct final token does not show whether hidden states preserve task distances and neighborhoods or whether an intervention composes correctly. Nonlinear and manifold methods replace straight lines with curved hidden-state paths between selected behaviors (Wurgaft et al., 2026; Oozeer et al., 2026), yet a set of paths need not define one transformation that acts consistently across the task domain or obeys composition and closure. Work that locates task geometry already present in activations relies on task-specific pipelines, such as clustering and projecting sparse-autoencoder features (Engels et al., 2025) or fitting Fourier and helical features (Zhou et al., 2024; Kantamneni & Tegmark, 2025). Inspired by group actions and geometric priors (Higgins et al., 2018; Bronstein et al., 2021), we instead make the task's metric, topology, and transformations explicit in a learned interface, using one encoder–decoder and one family of objectives across tasks with different geometries.

We introduce STRUCTURE-ALIGNED STEERING, which reconciles task relations with hidden-state and unembedding geometry through a learned *steerer*: an encoder maps hidden states into a geometric task realization, a task transformation acts there, and a decoder returns a state readable by the remaining layers. We first prescribe the task geometry. Adding calibrated weekday-token unembedding directions at an intermediate layer yields anchor states that make the model predict the corresponding weekdays but have neither the uniform one-day displacement required by vector

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

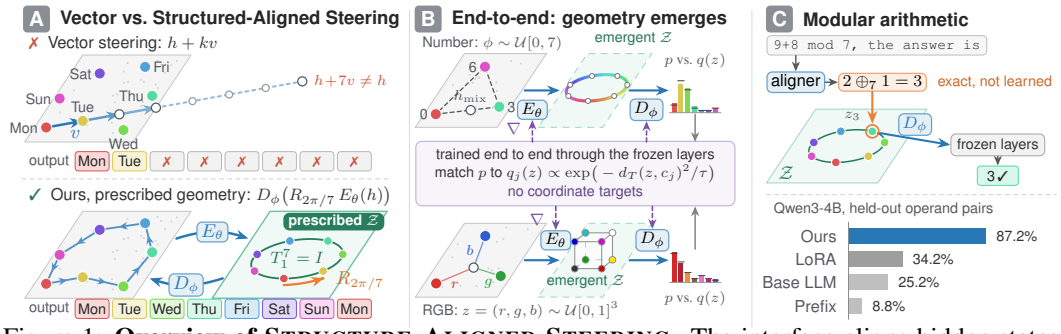


Figure 1: **Overview of STRUCTURE-ALIGNED STEERING.** The interface aligns hidden states with task structure so operations can be performed in task coordinates and decoded back into model-readable states. **(A)** Prescribed Weekday coordinates support cyclic composition and closure, unlike repeated vector addition. **(B)** Without coordinate targets, output supervision induces compatible Number-circle and RGB-cube realizations that support steering. **(C)** With an operand aligner, the same interface enables modular computation through the frozen model and outperforms parameter-matched PEFT baselines across eight models.

steering nor an equally spaced circular arrangement. We train the encoder to map anchors and their interpolations to prescribed circular coordinates, and the decoder to map them back. A $2\pi/7$ rotation then advances any weekday and closes after seven applications; compared with vector and manifold steering, only our steerer combines high path accuracy with perfect repeated-step and cycle return.

We next ask whether output feedback alone can teach this correspondence. We relabel the weekday phases as integers 0–6, mixing only three anchors, and then move to RGB colors. Training passes each decoded state through the remaining layers and matches the resulting candidate-token distribution, induced by hidden-state/unembedding geometry, to one determined by task-space relations, without specifying encoder coordinates. The Number steerer learns a compatible circular realization with a smooth 6–0 transition, while the RGB steerer organizes colors by cube neighborhoods and relative distances. Perturbing learned coordinates at inference or training against altered task relations reduces accuracy. Task structure thus defines only the output targets; its organization in learned coordinates emerges from matching task relations with hidden-state-to-output geometry.

Finally, modular arithmetic demonstrates practical value: combined with an operand aligner, the steerer lets the frozen model itself output the answer and outperforms logistic regression, layer-local LoRA, and prefix tuning across all eight models. In most models, its encoder also arranges interpolated residue phases along an ordered closed circle. We thus contribute one encoder–decoder interface that realizes four task structures, from a prescribed weekday cycle to the RGB cube and modular addition; mechanistic evidence that compatible realizations can emerge from output feedback alone; and a practical demonstration of computation through the learned interface.

2 RELATED WORK

Activation and representation steering. Activation and representation steering control outputs by modifying intermediate states rather than model parameters. Some methods extract contrastive directions or causal task representations (Merullo et al., 2024; Todd et al., 2024); others locate relevant heads (Li et al., 2023), learn low-rank (Wu et al., 2024) or adversarial (Zhang et al., 2024) edits, optimize control signals over the state space (Kong et al., 2024), target sparse autoencoder features (Chalnev et al., 2024), model concept subspaces (Postmus & Abreu, 2024), or condition interventions on the input (Lee et al., 2025). Yet whether using additions, projections, or learned maps, these interventions remain in hidden-state space, whose distances, interpolations, and composition rules need not match task structure.

Nonlinear and manifold steering. This limitation has motivated methods that explicitly model hidden-state geometry. One line shapes trajectories between selected behaviors: Angular (Vu & Nguyen, 2025) and Spherical (You et al., 2026) Steering use rotations, Curveball learns curved trajectories (Raval et al., 2026), Manifold Steering connects behavior centroids (Wurgaft et al., 2026), and Riemannian-Manifold Steering (Oozeer et al., 2026) and FishBack (Wang et al., 2026) derive paths or directions from pullback metrics. Another writes into geometric features the model already

represents, such as a circular weekday subspace (Engels et al., 2025). The first improves paths or local distortion without aligning interventions with a task's metric, topology, or reusable transformations; the second requires a native feature that not every task provides. **STRUCTURE-ALIGNED STEERING instead learns a single coordinate interface shared across tasks,** making continuous organization and repeated composition directly testable. Section I compares it with a circular-subspace intervention in the same weekday setting and at that method's own intervention site.

Structured computation and adaptation. Language-model representations can spontaneously develop meaningful geometry: spatial and temporal (Gurnee & Tegmark, 2024), syntactic (Diego-Simón et al., 2024), circular (Engels et al., 2025), prompt-dependent (Kirsanov et al., 2025), task-vector (Yang et al., 2025b), and arithmetic (Zhou et al., 2024; Kantamneni & Tegmark, 2025). This motivates us to ask whether output supervision can form an internal geometric realization of known task structure without assuming a matching pre-existing feature or supervising internal coordinates. Like LoRA and prefix tuning, our interface adds parameters around a frozen model (Hu et al., 2021; Li & Liang, 2021), but its inspectable geometry provides a mechanistic view of lightweight adaptation.

3 METHODOLOGY

3.1 TASK STRUCTURE AND HIDDEN-STATE MISMATCH

A common formalism for structured spaces. We describe hidden-state space and task space in one framework. For either space, let $\mathcal{A}$ denote its elements. A geometric realization $\mathbf{G} = (\mathcal{X}, \delta, \Omega)$ has a representation map $\rho : \mathcal{A} \to \mathcal{X}$, a metric δ defining distance, and a set Ω of transformations and combination operations involving those elements. An operation $\omega_{\mathcal{A}}$ on elements is represented by $\omega_{\mathcal{X}} \in \Omega$ if

$$\rho(\omega_{\mathcal{A}}(s_1, \ldots, s_r)) = \omega_{\mathcal{X}}(\rho(s_1), \ldots, \rho(s_r)), \qquad s_i \in \mathcal{A}, \tag{1}$$

so that representing an operation's result agrees with operating on its representations. Realizations are not unique: different coordinates can express the same semantic structure when their distances and operations correspond (formalized in Section A.1).

Hidden-state and task realizations. In hidden-state space, the element and coordinate domains coincide, $\mathcal{A}_H = \mathcal{X}_H = \mathbb{R}^d$, with $\rho_H(h) = h$ and structure $\mathbf{H} = (\mathbb{R}^d, \|\cdot\|_2, \Omega_H)$. Here Ω_H includes translations $\tau_v(h) = h + v$ and affine maps $A_{M,b}(h) = Mh + b$: vector steering uses the former, while representation-editing methods (Ravfogel et al., 2020; Belrose et al., 2023) use more general affine maps, including projections. These operations are available in $\mathbb{R}^d$ but have no intrinsic task-level meaning. Task space instead has elements $\mathcal{A}_T$, each paired with a designated output token, and realization $\mathbf{T} = (\mathcal{Z}, d_T, \Omega_T)$ with $c = \rho_T : \mathcal{A}_T \to \mathcal{Z}$; we call the $(\mathbf{T}, c)$ used for sampling and supervision the *reference realization*. For weekdays, $\mathcal{A}_T = \{\text{Monday}, \ldots, \text{Sunday}\}$ and $c(\text{day}_i) = (\cos\frac{2\pi i}{7}, \sin\frac{2\pi i}{7})$ on $\mathcal{Z} = S^1$, with chord distance d_T. Adjacent days are $2\sin(\pi/7)$ apart, and rotations $T_k(z) = R_{2\pi k/7} z$ realize day advances with $T_1^7 = I$. For RGB, $\mathcal{Z} = [0,1]^3$ with $d_T(z, z') = \|z - z'\|_2$, and Ω_T contains the basis synthesis $B_{\boldsymbol{\alpha}}(e_R, e_G, e_B) = (\alpha_R, \alpha_G, \alpha_B)$ for $\boldsymbol{\alpha} \in [0,1]^3$. In both cases, d_T makes semantic relations among task elements explicit.

Hidden-state and task structures need not agree. Algebraically, consider advancing a weekday one day at a time: vector steering repeats one translation, so $\tau_v^k(h) = h + kv$ never returns to h for $v \neq 0$, whereas the task's one-day rotation satisfies $T_1^7 = I$. Geometrically, distances between hidden states that make the model predict different task elements need not track d_T, and linear combinations of such states need not realize $B_{\boldsymbol{\alpha}}$: mixing states that make the model predict red and blue with $\boldsymbol{\alpha} = (1/2, 0, 1/2)$ need not make it predict purple. Vector steering that judges success only by a selected element's output can thus change an isolated output without reconciling either mismatch.

A coordinate interface reconciles the two spaces. We address both mismatches with an encoder $E_\theta : \mathcal{H} \to \mathbb{R}^m$ and a decoder $D_\phi : \mathbb{R}^m \to \mathcal{H}$, where $\mathcal{Z} \subset \mathbb{R}^m$ holds the reference coordinates; an equivalent realization serves equally, with ω_T expressed in its coordinates. Call a_s an output anchor for task element s if replacing the hidden state at the intervention layer with a_s makes the model predict s's token. If a task operation satisfies $\omega_T(c(s_1), \ldots, c(s_r)) = c(s')$, its hidden-state counterpart should map the anchors a_{s_i} to an anchor for s'. Operations applied directly in hidden-state space need not do so, as the red–blue example above shows, so the steerer routes the operation through task coordinates,

$$(a_{s_1}, \ldots, a_{s_r}) \longmapsto D_\phi(\omega_T(E_\theta(a_{s_1}), \ldots, E_\theta(a_{s_r}))), \tag{2}$$

where the encoder is trained to carry each anchor a_{s_i} to its reference coordinate $c(s_i)$, ω_T yields $c(s')$ there, and the decoder is trained to turn $c(s')$ back into an anchor for s', so the task operation acts on hidden states with its intended meaning and the two spaces are reconciled (Figure 1A).

3.2 Learning a readable coordinate interface

Unembedding directions construct output anchors. We now construct the output anchors required by the steerer. Let $h_{\ell,p}$ be the last-token state of a carrier prompt p at layer ℓ. We construct the anchor for task element s via

$$a_{s,p}(\alpha) = h_{\ell,p} + \alpha \bar{u}_s, \qquad \bar{u}_s = u_s / \|u_s\|_2. \tag{3}$$

where u_s is the unembedding of s. Writing the later blocks as additive residual updates Δ_k and the final normalization as Norm, token j then receives the logit $g_j(a_{s,p}(\alpha)) = u_j^\top \mathrm{Norm}\big(a_{s,p}(\alpha) + \sum_{k=\ell+1}^{L} \Delta_k\big)$. Because the residual stream carries the injected direction forward, increasing α raises the logit of s's token until it wins over the full vocabulary, while reducing the carrier's influence. We select α by this top-1 check (scale selection and sensitivity in Section B.3; carrier prompts in Section C) and write the resulting anchor as a_s. These anchors are output-readable but need not inherit the geometry of the task space.

Anchor mixtures train a nonlinear coordinate interface. Training the interface requires hidden states paired with task coordinates. For support elements $\mathcal{B} \subseteq \mathcal{A}_T$ with anchors a_s, each sampled coordinate $z \in \mathcal{Z}$ receives

$$h_{\mathrm{mix}}(z) = \sum_{s \in \mathcal{B}} \beta_s(z) a_s, \tag{4}$$

where the coefficient rule $\beta(z)$ expresses z through the support elements. These mixtures equal the anchors at support elements, $h_{\mathrm{mix}}(c(s)) = a_s$, and fill the coordinates in between, so training on them extends the conditions of Equation (2) to the whole task domain. The encoder and decoder are SiLU MLPs,

$$E_\theta : d \to 256 \to 256 \to m, \qquad D_\phi : m \to 256 \to 256 \to d, \tag{5}$$

where m is the dimension of the reference coordinates; the language model stays frozen. **The steerer is a unified interface: unlike task-specific pipelines that locate geometry in hidden states** (Engels et al., 2025; Zhou et al., 2024), only the reference realization, β, and the task operation vary by task.

Directly supervised coordinate geometry. We instantiate the interface on four tasks: **Weekday** and **Number** place seven elements on the circle, **RGB** uses the unit cube, and **Modular** (Section 4.3) adds addition modulo seven on the circle. We first test on Weekday whether the steerer can realize a prescribed circular geometry when the activation–coordinate correspondence is supplied directly. For the seven day anchors ($\mathcal{B} = \mathcal{A}_T$) and t on a uniform grid over $[0, 1)$, let $z = c_i(t)$ lie a fraction t along the arc from $c(\mathrm{day}_i)$ to the next day, with $h_{\mathrm{mix}}(z) = (1-t)a_{\mathrm{day}_i} + t\, a_{\mathrm{day}_{i+1}}$ by Equation (4). We minimize

$$\underbrace{\lambda_E \| \mathrm{unit}(E_\theta(h_{\mathrm{mix}}(z))) - z\|_2^2}_{\text{encoder loss}} + \underbrace{\lambda_D \|D_\phi(z) - h_{\mathrm{mix}}(z)\|_2^2}_{\text{decoder loss}}. \tag{6}$$

where $\mathrm{unit}(x) = x/\|x\|_2$ places the encoder output on the unit circle. The encoder thus maps each hidden-state segment between neighboring anchors onto its arc of the prescribed circle, and the decoder maps circle points back to states the model reads as weekdays. At inference, we replace the carrier state $h_{\ell,p}$ with $h_s^{(k)} = D_\phi(T_k(E_\theta(h_s^{(0)})))$, where $h_s^{(0)} = D_\phi(c(s))$ decodes the starting day s, so that the model predicts the day k steps later.

End-to-end emergent coordinate geometry. We next ask whether the steerer can learn such a geometry from end-to-end training alone on Number and RGB. The steerer receives no direct coordinate-target supervision to produce z (Figure 1B). Instead, we replace $h_{\ell,p}$ with $D_\phi(E_\theta(h_{\mathrm{mix}}(z)))$ and train the steerer so that the probabilities induced by this state — through the remaining layers $F_{\ell \to L,p}$ and inner products of the resulting final state with the unembedding — reflect where z lies in the task space, thereby reconciling the model's geometry with the task's. Comparing the two geometries requires task elements that are also tokens, so each task designates reference elements $\mathcal{Y} = \{s_j : j \in \mathcal{J}\}$ with single-token labels at coordinates $c_j = c(s_j)$, and we write $p_j^{\mathrm{full}}(z)$ for their post-intervention probabilities under the model's full-vocabulary softmax at coordinate z. We match them to

$$q_j(z) = \frac{\exp(-d_T(z, c_j)^2/\tau)}{\sum_{i \in \mathcal{J}} \exp(-d_T(z, c_i)^2/\tau)} \tag{7}$$

the probabilities over reference elements that a final state consistent with the task geometry should induce, and minimize

$$-\sum_{j\in\mathcal{J}} q_j(z)\log p_j^{\text{full}}(z). \tag{8}$$

This rewards outputs consistent with task distances and raises reference-token mass (Section B.5).

Number keeps Weekday's seven circular phases, relabels them $0,\dots,6$ as reference elements, and mixes only the anchors in $\mathcal{B}=\{0,3,6\}$: for $z=(\cos(2\pi\phi/7),\sin(2\pi\phi/7))$ with $\phi\sim\text{Uniform}[0,7)$, $\beta(z)$ interpolates linearly along the closed path $a_0\to a_3\to a_6\to a_0$. This tightens the Weekday setting in two ways: weekday output tokens are removed, so success cannot come from reusing them, and three anchors must support seven elements, far sparser hidden-state support than Weekday's seven. RGB then leaves their shared circle for a three-dimensional cube, a harder structure to realize: $\mathcal{B}=\{\text{red},\text{green},\text{blue}\}$, $z=(r,g,b)$ sampled uniformly from $[0,1]^3$, and $h_{\text{mix}}(z)=ra_{\text{red}}+ga_{\text{green}}+ba_{\text{blue}}$. Its 64 reference elements are named colors at their RGB coordinates, with the first token of multi-token names as the output label (carriers and element sets in Sections C.3 and C.4; the same objective also applies to Weekday, Section D.4).

The interface family can approximate an ideal cross-space computation. The preceding supervision asks the steerer to encode task-bearing hidden states as coordinates organized by the task geometry and to decode operated coordinates into model-readable states. The following result shows that its SiLU encoder–decoder architecture has sufficient capacity to approximate both mappings.

Theorem 1 (informal)**.** *Suppose continuous ideal maps $E^\star$ and $D^\star$ encode a compact set of relevant hidden states into a compact set of task coordinates and decode operated coordinates back to hidden states. Write Π for the output map that sends a layer-ℓ state to its full-vocabulary distribution, namely $F_{\ell\to L,p}$ followed by the unembedding and softmax. If $D^\star$ and Π are Lipschitz, and the task operation ω_T is Lipschitz on a neighborhood of those coordinates, then for any encoder tolerance ϵ_E fixing that neighborhood and any decoder tolerance ϵ_D, sufficiently wide two-hidden-layer SiLU MLPs approximate $E^\star$ and $D^\star$ uniformly within ϵ_E and ϵ_D. After ω_T, their output distribution differs from the ideal one by at most $L_\Pi(\epsilon_D+L_DL_\omega\epsilon_E)$.*

Thus, whenever an ideal continuous interface exists, a sufficiently wide steerer can approximate both its encoder–decoder maps and the output distribution it produces after a task operation. The theorem establishes this representational capacity and the following experiments test whether training realizes it (Section J). See Section A.3 for the formal statement and proof.

4 Experiments

Models and intervention setting. We evaluate Llama-3-8B/70B and Llama-3.1-8B (Grattafiori et al., 2024), Llama-3.2-3B (Meta AI, 2024), together with Qwen3-1.7B/4B/8B/14B (Yang et al., 2025a); unless noted, main-text results use Qwen3-4B. Steerers replace the final prompt-token state at the layer nearest three-quarters of model depth. Model-specific layers, training and evaluation details, prompt templates, and remaining replications are in the section-matched appendices and Sections B and C.

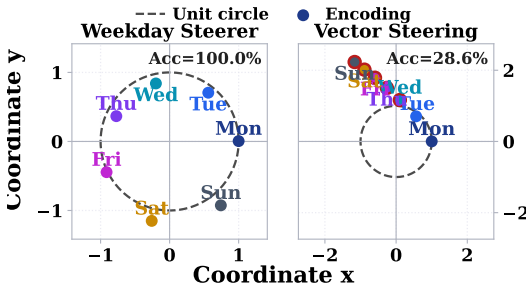


Figure 2: The Weekday steerer organizes the seven weekday states around the prescribed cycle, whereas repeated addition of the Tuesday-minus-Monday vector follows an open trajectory and is correct only for Monday and Tuesday.

4.1 The steerer realizes cyclic composition beyond vector steering

The Weekday steerer closes the cycle; vector steering does not. We first confirm that the Weekday steerer fits its prescribed coordinates: after a common rescaling, the encoded weekday anchor states lie at approximately equal intervals around the circle in Figure 2 (left), as required by their training targets. The substantive test is compositional. Starting from the reconstructed Monday state, we repeatedly apply encoding, a $2\pi/7$ rotation, and decoding; the model predicts every weekday correctly and returns to Monday after the seventh application. Vector steering has no analogous cyclic compositionality. We set $v_{\text{day}}=a_{\text{Tuesday}}-a_{\text{Monday}}$ and evaluate the repeated

translations $h_{\text{vec}}^{(j)} = a_{\text{Monday}} + jv_{\text{day}}$, $j = 0, \ldots, 7$, over the same traversal. In Figure 2 (right), only Monday and Tuesday are correct. Thus, even in a setting favorable to vector steering, where its endpoints are known and its first transition succeeds, a translation fitted to one pair cannot reproduce the task's cyclic composition law. See Section D.2 for other models.

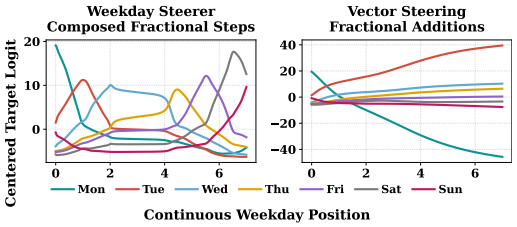


Figure 3: Composing 0.01-day transformations around the complete cycle transfers logit dominance smoothly between neighboring weekdays and returns from Sunday to Monday; fractional vector additions do not close the cycle. See Section D.2 for other models.

Fractional transformations compose into a continuous cycle. The seven endpoint transitions in Figure 2 establish discrete periodicity, but integer-day advances or retreats do not test continuous composition. We therefore divide each one-day transformation into $N = 100$ steps and use the recurrence $h^{(n+1)} = D_\phi\big(R_{2\pi/(7N)} E_\theta(h^{(n)})\big)$, so 100 successive 0.01-day transformations must equal one one-day transformation, including Sunday back to Monday. Figure 3 shows that the Weekday steerer composes these transformations throughout the cycle: neighboring-weekday logits exchange dominance smoothly in semantic order and close across the Sunday–Monday boundary. In contrast, fractional vector composition $h_{\text{vec}}^{(n+1)} = h_{\text{vec}}^{(n)} + v_{\text{day}}/N$ continues along a line and remains concentrated near its first target. Our steerer thus realizes a reusable continuous cyclic action rather than a displacement tied to one source–target pair: its task-space operations remain coherent through repeated decoding into hidden states and propagation to output logits, reconciling the two spaces.

Only the Weekday steerer turns a successful path into a reusable cyclic operation. We compare our steerer with Manifold Steering (Wurgaft et al., 2026) on Llama-3.1-8B layer 28. Its corpus contains all $7 \times 7 = 49$ prompts `Q: What day is {number} days after {weekday}? A:`, from which we sample 16 carriers. Manifold Steering fits PCA-64, with components U and coordinates $P(h)$, then a periodic spline γ through weekday centroids. We test ordered intermediate decoding and a shared day-advance map that closes after seven uses. For each of the $\binom{7}{2} = 21$ unordered weekday pairs (s, t), we sample 50 positions on $h_{s\to t}^{\text{ours}}(u) = D_\phi(c_{s\to t}(u))$ and $h_{s\to t,p}^{\text{MS}}(u) = h_p + U^\top[\gamma_{s\to t}(u) - P(h_p)]$, where $c_{s\to t}$ and $\gamma_{s\to t}$ are the shortest circular and spline arcs. After running the remaining layers, final-target accuracy checks whether $u = 1$ outputs t, while ordered-path accuracy compares each waypoint with the nearest weekday on $c_{s\to t}$; both are averaged across carriers. For composition, Ours applies $T(h) = D_\phi(R_{2\pi/7} E_\theta(h))$ repeatedly, whereas Manifold Steering advances the nearest spline coordinate by its fixed Monday-to-Tuesday increment before lifting it to hidden space. From Monday, repeated-step accuracy scores Monday, Tuesday, . . ., Sunday, Monday, while cycle return scores only the last step. See Section H for definitions. Both methods reach every target, but Table 1 shows that Ours better preserves weekday order and realizes the reusable cyclic operation. Thus a nonlinear path alone is insufficient to capture the task space's semantic structure. Our steerer also outperforms the circular-subspace intervention of Engels et al. (2025) (Section I).

Table 1: Comparison with Manifold Steering in the weekday setting: our steerer better preserves weekday order and supports repeated cyclic composition.

Method	Final Target	Ordered Path	Repeated Step	Cycle Return
Ours	1.000	**0.934** ↑.059	**1.000** ↑.625	**1.000** ↑1.000
Manifold	1.000	0.875	0.375	0.000

4.2 Output supervision induces structured coordinate realizations

Number learns a circular realization without coordinate supervision. The Weekday experiments showed that the steerer realizes a circular task structure when the activation–coordinate correspondence is supplied directly. We now let an internal realization emerge from output supervision, retaining seven reference phases but relabeling them $0, \ldots, 6$. Number constructs mixtures only along the sparse path $a_0 \to a_3 \to a_6 \to a_0$, providing substantially less hidden-state support than Weekday. The steerer is trained only through the geometry-derived out-

Table 2: Across five seeds the Number steerer predicts the nearest digit over densely sampled circular phases more accurately than the direct hidden-state mixture or a linear-regression baseline.

Steerer	Mixture	Regr.
97.9±1.2%	41.8±1.6%	17.4±3.5%

put objective in Equations (7) and (8), without coordinate targets. For each seed, we sample 512 phases, construct $h_{\text{mix}}(z)$ using the three-anchor instantiation of Equation (4), and score whether the predicted digit is the reference element nearest the sampled phase. Across five seeds, the steerer obtains 97.9% mean accuracy. For comparison, the direct hidden-state mixture inserts $h_{\text{mix}}(z)$ at layer ℓ, testing whether straight-line mixtures already realize the circle. A linear-regression baseline fits an affine map from an independent set of 512 mixtures to circular coordinates, using coordinate labels unavailable to the steerer, and predicts the nearest-phase digit (formulation and fitting procedure in Section E.1); it tests whether a linear map can replace the learned interface on the same inputs. They obtain mean accuracies of 41.8% and 17.4%, respectively, in Table 2. Exact anchor decoding is 100%, so remaining errors occur between anchors. Thus output-level training induces a compatible realization of the shared circular structure from sparse mixtures rather than relying on weekday-specific output tokens. The same pattern holds across models (Sections E.3, E.5 and E.6).

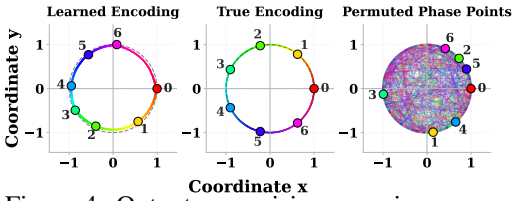


Figure 4: Output supervision organizes sparse hidden mixtures into a continuous circle with the correct phase neighborhoods, up to the learned chart's global orientation; permuting phase positions destroys this correspondence. See Section E.3 for other models.

Sparse hidden-state support unfolds into a continuous number circle. To test whether the Number steerer learns the task's cyclic geometry beyond output-level accuracy, we densely sample 100 phases between each neighboring pair, construct $h_{\text{mix}}(z)$ using the same three-anchor rule, encode these states, and align the first encoded phase to its reference position for visualization. In Figure 4, the learned path follows the full circle, preserves neighboring phases, and places phases near 6 and 0 next to one another; its global orientation may reverse. For comparison, we construct a distribution-matched random baseline by permuting the same circle coordinates across sampled phases, preserving the circular point distribution while destroying phase order and neighborhoods. The encoder thus organizes the full interpolated path, not just the seven reference locations, into the task's cyclic geometry, recovered after one global alignment (alignment procedure in Section E.2).

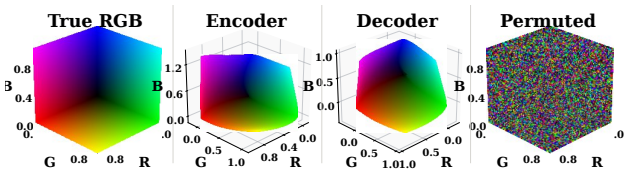
RGB extends output-supervised alignment to three dimensions. Weekday prescribes its circle and Number recovers one from output supervision alone; we next test whether such supervision can induce a three-dimensional RGB realization. We sample $z = (r, g, b) \in [0, 1]^3$, construct $h_{\text{mix}}(z)$ from the red, green, and blue output anchors as specified in Equation (4), and train through the distance-based output target in Equation (7); the encoder never receives z as a target. We evaluate 64 representative RGB coordinates, each paired with its corresponding named-color target. Across five training seeds, the Qwen3-4B RGB steerer achieves 70.3% mean accuracy. For comparison, the direct hidden-state mixture replaces the original layer-ℓ state with the same raw mixture $ra_{\text{red}} + ga_{\text{green}} + ba_{\text{blue}}$, testing whether linear hidden-state mixtures already realize RGB relations.

Table 3: Across five training seeds, the RGB steerer predicts the correct named-color token more accurately than the direct hidden-state mixture and a linear color probe. See Section F.5 for other models.

Steerer	Mixture	Probe
70.3±9.8%	4.7±0.0%	9.4±0.0%

We additionally fit a linear color probe from the layer-ℓ state of the final color-name token in "This color is *name*" to the RGB coordinates of 32 colors, evaluated on the other 32 (Section F.1). Both baselines obtain 4.7% and 9.4% accuracy in Table 3. Both hidden-state tests fail: the mixture does not yield intermediate colors, and the model's color-name states expose no linearly readable RGB geometry at any layer. Nor does the unembedding supply it. Expressing each color token's unembedding in the red, green, and blue basis yields coordinates whose pairwise distances correlate with RGB distances at only $r = 0.23$–0.24 across eight models (Section F.2), so a red–blue mixture need not move toward purple. The geometry must therefore come from a learned interface mapping RGB coordinates to model-readable states (Section F.3). Its advantage over both baselines holds across models, insertion layers, and colors held out from training (Sections F.5, F.6 and F.10).

The interface carries RGB structure in both directions. Discrete-color accuracy does not show whether the steerer forms a continuous RGB realization or returns it to model-readable states. We sample a 64×64 grid on each of three cube faces, form $h_{\text{mix}}(z) = ra_{\text{red}} + ga_{\text{green}} + ba_{\text{blue}}$ for each $z = (r, g, b)$, and examine both ends of the interface in Figure 5.



MSE vs. true: encoder 0.0077, decoder 0.0233, permuted 0.2256 ± 0.0013

Figure 5: Output supervision induces RGB-consistent geometry at both ends of the interface: aligned encoder coordinates reconstruct the cube faces, while the decoder also preserves RGB structure, both visibly better than random permutation. See Section F.7 for other models.

For the encoder, we compute $E_\theta(h_{\text{mix}}(z))$ and apply one affine alignment, fitted only on the encoded red, green, blue, and black mixtures, to fix origin and basis conventions (alignment details in Section F.4). The aligned encodings form a slightly rounded cube whose faces preserve color gradients and fit the true RGB geometry significantly better than permuted RGB points ($p \ll 10^{-3}$; 2D face views in Section F.8). For the decoder, we pass each mixture through the full steerer, take the displacement $D_\phi(E_\theta(h_{\text{mix}}(z))) - h_{\ell,p}$, and express it in the basis of anchor displacements $D_\phi(E_\theta(a_i)) - h_{\ell,p}$ for $i \in \{\text{red}, \text{green}, \text{blue}\}$, so the coefficients measure movement along model-readable color directions. The decoded arrangement is visibly sheared, compressing the red–blue face, but preserves each face's color order and neighborhoods, with MSE 0.0233 against the same null. Since the raw mixtures are unreadable (4.7% above), the decoded states differ from their inputs while retaining their organization.

The learned encoding preserves global RGB distances. The visualization above samples only three cube faces, so it cannot show whether the steerer has learned the three-dimensional RGB geometry as a whole. We therefore test whether the encoder recovers the metric d_T that defines this geometry and, through Equation (7), its training targets. We encode the mixtures $h_{\text{mix}}(c_j)$ of all 64 representative colors, which lie throughout the cube, and compare the 2,016 pairwise Euclidean distances between their encodings with the corresponding distances $d_T(c_j, c_k)$, fitting a single global scale and no alignment. Figure 6 shows strong agreement (Pearson $r = 0.808$, Spearman $\rho = 0.817$, relative MSE 0.063). Since the encoder never receives coordinates and is supervised only through output distributions derived from d_T, output-level training induces an approximately metric-preserving realization of the whole cube, extending the continuous neighborhoods visible on its sampled faces.

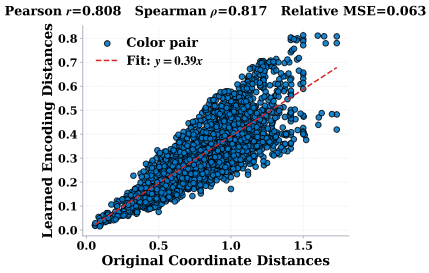


Figure 6: Across all pairs of 64 named colors, encoded distances strongly track RGB-space distances after one global rescaling (Pearson $r = 0.808$). See Section F.9 for other models.

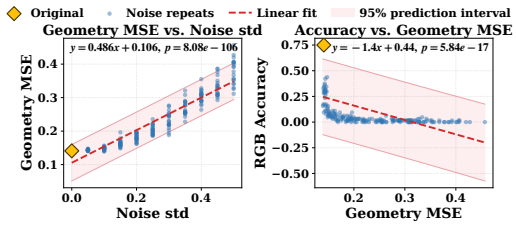


Figure 7: Adding Gaussian noise to the learned encoder coordinates increases their RGB-space error and sharply reduces full-vocabulary accuracy over the 64 representative colors from the unperturbed standard run's 75.0% (five-seed mean 70.3 ± 9.8%; Table 3) to nearly zero at $\sigma = 0.5$, linking geometric degradation directly to steering failure. See Section F.11 for other models.

Perturbing learned coordinates degrades RGB steering. To test whether the learned RGB structure contributes to steering, we add isotropic Gaussian noise with standard deviation $\sigma \in [0, 0.5]$ to each encoded color coordinate before decoding at inference, leaving the steerer and frozen model unchanged. For each noise level, we evaluate all 64 representative colors over 20 independent perturbations. In Figure 7, increasing σ raises coordinate error and sharply degrades output accuracy: Qwen3-4B falls from 75.0% without noise to 31.4% at $\sigma = 0.05$ and nearly zero at $\sigma = 0.5$. Thus the learned coordinate geometry is functionally important rather than merely visually regular.

Task-consistent geometry, not arbitrary latent geometry, supports steering. Having found that inference-time coordinate perturbations degrade steering, we next alter the geometry used in training while holding the other elements fixed. A cubic warp $(r, g, b) \mapsto (r^3, g^3, b^3)$ changes the distances defining $q(z)$, while two- and one-dimensional PCA projections discard degrees of freedom and cannot losslessly represent the RGB cube (Section A.5); each control is rescaled to match true RGB's mean pairwise distance, keeping the soft-target temperature at the same effective scale. Using the pairwise-distance probe from Figure 6,

Table 4 shows that every control reduces continuous accuracy and distance correlation, with the one-dimensional projection approaching chance. Thus successful steering depends on preserving the task's dimension and metric relations. See Section F.12 for other models.

4.3 Modular arithmetic: structured computation through the frozen model

Table 4: Task-consistent geometry best preserves color accuracy and RGB distances.

Training geometry	Cont. acc. (%)	Dist. corr.
True RGB (3D)	**62.7**	**0.808**
Cubic warp (3D)	26.7 ↓36.0	0.752 ↓.056
PCA-2D	27.8 ↓34.9	0.702 ↓.106
PCA-1D	6.5 ↓56.2	0.579 ↓.229

A steerer–aligner system solves modular arithmetic through the frozen model. After studying how the steerer represents and uses task geometry, we test its practical value on modular arithmetic, which is closely related to Number. Addition modulo seven forms the finite cyclic group $(\mathbb{Z}_7, \oplus)$, where $a \oplus b = (a + b) \bmod 7$ and $c(a \oplus b) = R_{2\pi b/7} c(a)$ under $c(j) = (\cos(2\pi j/7), \sin(2\pi j/7))$. Despite this simple structure, transformers can require prolonged grokking or specialized representations and objectives to generalize modular arithmetic (Power et al., 2022; Nanda et al., 2023; Saxena et al., 2024). We sample $a, b \in \{0, \ldots, 29\}$, query $p(a, b) =$ `{a}+{b} mod 7, the answer is` , and target $(a + b) \bmod 7$. Operand pairs are split into disjoint training and test sets (Section C.6). We train a seven-residue circular steerer and freeze it. An aligner then reads the layer-ℓ state with two role-specific branches, each supervised to predict one operand; their distributions are folded modulo seven and combined by exact circular convolution to select the decoder phase (Section C.7). The resulting $824d$-parameter system reaches 87.2% accuracy in Figure 8.

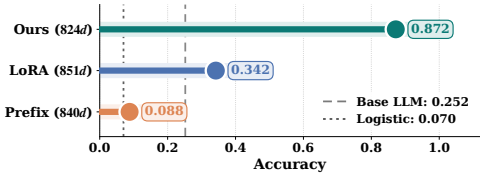


Figure 8: On Qwen3-4B, the steerer–aligner outperforms parameter-matched layer-local LoRA and prefix tuning on pair-disjoint modular queries. See Sections G.2 and G.9 for cross-model and 16-shot results.

The system outperforms the parameter-matched PEFT methods, layer-local LoRA ($851d$) and prefix tuning ($840d$), and the logistic-regression and unmodified-LLM baselines; the advantage holds across all eight models (Section G.2; budgets and output-interface controls in Sections G.1 and G.4). Its encoder also arranges interpolated residue phases along an ordered closed circle in most models, as the Number steerer does (Figure 4 and Section G.10), indicating that it learns the task's circular structure, not particular output tokens. Modulus, operand-range, seed, and layer sweeps confirm these results, while unseen prompt templates reduce accuracy (Sections G.5 and G.6). Thus the steerer both computes through the frozen model and learns the task's geometry.

The selected result coordinate supports a discrete group action. Accurate modular outputs do not show whether the selected coordinate remains usable for a subsequent task operation. For each held-out query, we therefore shift its predicted residue endpoint by balanced offsets $c + d \in \mathbb{Z}_7$ and decode the target $(a + b + c + d) \bmod 7$. Table 5 shows unchanged accuracy, while starting from the oracle result remains perfect. Thus the interface supports an additional discrete group action without introducing new errors; this endpoint test does not evaluate continuous rotation (Sections G.8 and G.9).

Table 5: An additional $\mathbb{Z}_7$ action preserves zero-shot modular accuracy.

Original	After $\mathbb{Z}_7$	Oracle
.872	.872	1.000

5 Conclusion

We introduced Structure-Aligned Steering, an interface that organizes hidden states into a geometric realization of supplied task structure, applies transformations there, and returns model-readable states. Weekday steering shows that this separation supports continuous cyclic composition beyond vector addition and a fitted nonlinear path. Number and RGB show that output supervision can learn a compatible internal realization without direct coordinate targets, while perturbation and wrong-geometry controls link that organization to behavior. Modular arithmetic demonstrates practical use: explicit operand prediction, exact circular composition, and model-readable write-back let the frozen model outperform matched adaptation baselines. Within known task structure, the method turns isolated output interventions into reusable, task-aligned computation.

AI USE STATEMENT

Generative AI tools assisted with refining hypotheses and terminology; designing, implementing, and debugging experiments; organizing result artifacts; producing visualizations; inspecting executable configurations; and drafting and editing the manuscript. The authors reviewed the resulting code and prose, checked metric definitions and reported values against saved outputs, verified prompt and split construction, and remain responsible for all claims, citations, and artifacts. No generative model was used to assign ground-truth labels or to judge the primary experimental outcomes.

REFERENCES

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. LEACE: Perfect linear concept erasure in closed form. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/d066d21c619d0a78c5b557fa3291a8f4-Abstract-Conference.html.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021. URL https://arxiv.org/abs/2104.13478.

Sviatoslav Chalnev, Matthew Siu, and Arthur Conmy. Improving steering vectors by targeting sparse autoencoder features, 2024. URL https://arxiv.org/abs/2411.02193.

Pablo Diego-Simón, Stéphane D'Ascoli, Emmanuel Chemla, Yair Lakretz, and Jean-Rémi King. A polar coordinate system represents syntax in large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/be36e50757bf9cd280aa74f89a7d1c23-Abstract-Conference.html.

Joshua Engels, Eric J. Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=d63a4AM4hb.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Wes Gurnee and Max Tegmark. Language models represent space and time, 2024. URL https://arxiv.org/abs/2310.02207.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations, 2018. URL https://arxiv.org/abs/1812.02230.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition, 2025. URL https://arxiv.org/abs/2502.00873.

Artem Kirsanov, Chi-Ning Chou, Kyunghyun Cho, and SueYeon Chung. The geometry of prompting: Unveiling distinct mechanisms of task adaptation in language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1855–1888, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. URL https://aclanthology.org/2025.findings-naacl.100/.

Lingkai Kong, Haorui Wang, Wenhao Mu, Yuanqi Du, Yuchen Zhuang, Yifei Zhou, Yue Song, Rongzhi Zhang, Kai Wang, and Chao Zhang. Aligning large language models with representation editing: A control perspective. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/41bba7b0f5c81e789a20bb16a370aeeb-Abstract-Conference.html.

Bruce W. Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e2dd53601de57c773343a7cdf09fae1c-Abstract-Conference.html.

Moshe Leshno, Vladimir Ya. Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. doi: 10.1016/S0893-6080(05)80131-5.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *Advances in Neural Information Processing Systems*, volume 36, pp. 41451–41530, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/81b8390039b7302c909cb769f8b6cd93-Abstract-Conference.html.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.353. URL https://aclanthology.org/2021.acl-long.353/.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets, 2024. URL https://arxiv.org/abs/2310.06824.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Language models implement simple word2vec-style vector arithmetic, 2024. URL https://arxiv.org/abs/2305.16130.

Meta AI. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models, 2024. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/. Model release announcement.

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.

Narmeen Oozeer, Shivam Raval, Philip Quirke, Manikandan Ravikiran, Jeff Phillips, Shriyash Upadhyay, and Amirali Abdullah. Riemannian-manifold steering: Geometry-aware generative autoencoders for label-free steering, 2026. URL https://arxiv.org/abs/2605.24942.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering Llama 2 via contrastive activation addition, 2023. URL https://arxiv.org/abs/2312.06681.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models, 2023. URL https://arxiv.org/abs/2311.03658.

Joris Postmus and Steven Abreu. Steering large language models using conceptors: Improving addition-based activation engineering, 2024. URL https://arxiv.org/abs/2410.16314.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022. URL https://arxiv.org/abs/2201.02177.

Shivam Raval, Hae Jin Song, Linlin Wu, Abir Harrasse, Jeff M. Phillips, Fazl Barez, and Amirali Abdullah. Curveball steering: The right direction to steer isn't always linear, 2026. URL https://arxiv.org/abs/2603.09313.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7237–7256, 2020. doi: 10.18653/v1/2020.acl-main.647. URL https://aclanthology.org/2020.acl-main.647/.

Eshika Saxena, Alberto Alfarano, François Charton, Zeyuan Allen-Zhu, Emily Wenger, and Kristin Lauter. Making hard problems easier with custom data distributions and loss regularization: A case study in modular arithmetic, 2024. URL https://arxiv.org/abs/2410.03569.

Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models, 2024. URL https://arxiv.org/abs/2310.15213.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering, 2023. URL https://arxiv.org/abs/2308.10248.

Hieu M. Vu and Tan M. Nguyen. Angular steering: Behavior control via rotation in activation space. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b0223cad0e73b793f31eb6cc41cefceb-Abstract-Conference.html.

Sihan Wang, Jiayi Zhao, Qingyan Cao, Hongbo Yao, and Lin Shu. Fishback: Pullback fisher geometry for optimal activation steering in transformers, 2026. URL https://arxiv.org/abs/2605.17231.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation finetuning for language models, 2024. URL https://arxiv.org/abs/2404.03592.

Daniel Wurgaft, Can Rager, Matthew Kowal, Vasudev Shyam, Sheridan Feucht, Usha Bhalla, Tal Haklay, Eric Bigelow, Raphael Sarfati, Thomas McGrath, Owen Lewis, Jack Merullo, Noah Goodman, Thomas Fel, Atticus Geiger, and Ekdeep Singh Lubana. Manifold steering reveals the shared geometry of neural network representation and behavior, 2026. URL https://arxiv.org/abs/2605.05115.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, et al. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Haolin Yang, Hakaze Cho, Yiqiao Zhong, and Naoya Inoue. Unifying attention heads and task vectors via hidden state geometry in in-context learning, 2025b. URL https://arxiv.org/abs/2505.18752.

Zejia You, Chunyuan Deng, and Hanjie Chen. Spherical steering: Geometry-aware activation rotation for language models, 2026. URL https://arxiv.org/abs/2602.08169.

Yihao Zhang, Zeming Wei, Jun Sun, and Meng Sun. Adversarial representation engineering: A general model editing framework for large language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/e4630f7c0660d944c132455c124e7d90-Abstract-Conference.html.

Tianyi Zhou, Deqing Fu, Vatsal Sharan, and Robin Jia. Pre-trained large language models use fourier features to compute addition. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/2cc8dc30e52798b27d37b795cc153310-Abstract-Conference.html.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to AI transparency, 2023. URL https://arxiv.org/abs/2310.01405.

A Formal Guarantees for Structured Interfaces

The main text argues that output supervision can induce a geometric realization consistent with task relations and that the resulting interface can support reusable operations. Formalizing these claims first requires separating task structure from an arbitrary internal coordinate chart, since output behavior need not identify the latter. We therefore begin by defining equivalent realizations and the geometry observable in the output distribution. We then show that distance-based training targets identify sampled coordinates in any equivalent realization used as the reference realization, and ask whether the SiLU function class can represent the required cross-space maps, and whether round-trip accuracy makes the induced hidden-state operations compose like their task-space counterparts. Finally, we identify dimensional settings in which no lossless continuous interface can exist.

A.1 Equivalent realizations and observable output geometry

Output-supervised training constrains the interface only through its decoded behavior, so different internal coordinate charts can be observationally indistinguishable. Before analyzing geometric recovery, we must distinguish such reparameterizations from changes to the task's semantic distances or operations. As in Section 3, a task realization $\mathbf{T} = (\mathcal{Z}, d_T, \Omega_T)$ represents elements through $c : \mathcal{A}_T \to \mathcal{Z}$. The chosen coordinates are not themselves identifiable. Let $E : \mathcal{M} \to \mathcal{Z}$ and $D : \mathcal{Z} \to \mathcal{M}$ be an interface on a task-bearing hidden-state set $\mathcal{M} \subseteq \mathcal{H}$, the subset of the hidden-state space of Section 3 that the interface is trained on. For a homeomorphism $\psi : \mathcal{Z} \to \mathcal{Z}'$, define $E' = \psi \circ E$ and $D' = D \circ \psi^{-1}$. Then $D'E' = DE$ on $\mathcal{M}$. The same task structure is expressed in the new coordinates by transporting its metric and operations:

$$\begin{aligned} d'_T(\psi(z), \psi(z')) &= d_T(z, z'), \\ \omega_\psi(z'_1, \dots, z'_r) &= \psi\big(\omega_T(\psi^{-1} z'_1, \dots, \psi^{-1} z'_r)\big) . \end{aligned} \tag{9}$$

Thus output behavior cannot distinguish coordinate charts related in this way. This equivalence requires transporting the structure, not merely transforming the points. For example, after a nonlinear change $u = \psi(z)$, the equivalent metric is $d'_T(u, v) = d_T(\psi^{-1}(u), \psi^{-1}(v))$; continuing instead to use the ordinary Euclidean distance $\|u - v\|_2$ generally changes pairwise distances and neighborhoods. The same distinction holds for operations through Equation (9). The display alignments used for the Number encoding in Figure 4 and the RGB encoder and decoder visualizations in Figure 5 are fitted only after training to overlay a learned chart with the reference coordinates. They do not alter the training targets, decoded outputs, or task operations; the separate distance and action diagnostics test whether structure is preserved beyond the visual alignment. By contrast, the wrong-geometry controls in Table 4 and Section E.4 replace the training coordinates while retaining the original distance or operation rule (or collapse dimensions noninvertibly). They therefore change the semantic relations encoded by the targets rather than re-expressing the same relations in another chart.

Although output supervision does not select a unique internal chart, Equation (8) directly constrains the probability pattern produced after decoding: it rewards geometry-derived relative probabilities over the reference tokens and, through full-vocabulary normalization, their total mass against background tokens. Extending the notation $p_j^{\mathrm{full}}(z)$ from Equation (8), let $p_v^{\mathrm{full}}(h)$ denote the full-vocabulary probability of token v after an intervened state h is processed by those layers. Evaluating it at $h = D(E(h_{\mathrm{mix}}(z)))$ recovers the earlier notation: $p_{s_j}^{\mathrm{full}}(h) = p_j^{\mathrm{full}}(z)$. For the reference-token set $\mathcal{Y} = \{s_j : j \in \mathcal{J}\}$, define its total mass and conditional distribution by

$$M_{\mathcal{Y}}(h) = \sum_{j \in \mathcal{J}} p_{s_j}^{\mathrm{full}}(h), \qquad \bar{p}_j(h) = \frac{p_{s_j}^{\mathrm{full}}(h)}{M_{\mathcal{Y}}(h)}. \tag{10}$$

The first quantity measures whether the intervention suppresses background tokens; the second measures the relative geometry expressed among task elements.

A.2 Geometry identified by the soft-target objective

A task structure admits multiple equivalent geometric realizations, any of which may serve as the reference realization defined in Section 3; as shown above, the steerer may likewise use any compatible internal chart. The relevant question is therefore whether the loss remains informative

whichever equivalent realization is used to sample z and construct the targets. Its relative probabilities encode distances from z to the reference elements. For the Euclidean Number and RGB reference realizations, we show that these distances recover z; transporting the coordinates and metric together gives the same guarantee when an equivalent realization is used as the reference, without requiring the steerer's internal chart to equal it.

Theorem 2 (Geometric information in the output objective)**.** *Let $\mathcal{Z} \subseteq \mathbb{R}^m$, let d_T be Euclidean distance, and suppose the reference coordinates $c_0, \dots, c_J$ affinely span $\mathbb{R}^m$. Let $C \in \mathbb{R}^{J\times m}$ have rows $(c_j - c_0)^\top$, let $\Delta_j = \|c_j\|_2^2 - \|c_0\|_2^2$, and set $\ell_j(z) = \log(q_j(z)/q_0(z))$. Then the geometry-derived output target recovers every sampled reference coordinate as*

$$z = \frac{1}{2} C^\dagger \big(\tau \ell(z) + \Delta\big), \tag{11}$$

so $z \mapsto q(z)$ is injective. Moreover, for any full-vocabulary distribution p^{full} with positive reference-token probabilities, let $M_{\mathcal{Y}} = \sum_{j\in\mathcal{J}} p_j^{\text{full}}$ and $\bar{p}_j = p_j^{\text{full}}/M_{\mathcal{Y}}$. Its soft-target cross-entropy decomposes as

$$-\sum_{j\in\mathcal{J}} q_j(z) \log p_j^{\text{full}} = H(q(z)) + \mathrm{KL}(q(z)\|\bar{p}) - \log M_{\mathcal{Y}}. \tag{12}$$

Consequently, if this cross-entropy is at most $H(q(z)) + \epsilon$, then $M_{\mathcal{Y}} \geq e^{-\epsilon}$ and $\|\bar{p} - q(z)\|_1 \leq \sqrt{2\epsilon}$. If ψ defines an equivalent realization as in Equation (9), with $c'_j = \psi(c_j)$ and the transported metric d'_T, then its target satisfies $q'_j(\psi(z)) = q_j(z)$ and therefore identifies the transformed coordinate $\psi(z)$ as well.

This theorem formalizes why the output-level objective can supervise more than the identity of a discrete label. In Equation (11), the log-probability ratios $\ell(z)$ encode the distance relations needed to reconstruct every sampled z. The identity $q'_j(\psi(z)) = q_j(z)$ makes this guarantee equivariant to the choice of reference realization: jointly transporting the sampled points, reference elements, and metric reconstructs $\psi(z)$ while preserving the same task structure. The loss therefore supplies continuous geometric information in any such realization but does not select the internal chart implemented by the steerer. In Equation (12), the excess over $H(q(z))$ is the sum of $\mathrm{KL}(q(z)\|\bar{p})$ and $-\log M_{\mathcal{Y}}$; reducing the loss therefore moves the normalized reference-token distribution toward $q(z)$ while moving its total full-vocabulary mass toward one.

Proof of Theorem 2. Choose c_0 as a reference and let $C \in \mathbb{R}^{J\times m}$ have rows $(c_j - c_0)^\top$ for $j = 1, \dots, J$. Affine spanning of the reference coordinates is equivalent to $\mathrm{rank}(C) = m$. Expanding the squared distances in Equation (7) gives

$$\ell_j(z) := \log \frac{q_j(z)}{q_0(z)} = \frac{2(c_j - c_0)^\top z - (\|c_j\|_2^2 - \|c_0\|_2^2)}{\tau}. \tag{13}$$

Writing $\ell(z) = (\ell_1(z), \dots, \ell_J(z))^\top$ and $\Delta_j = \|c_j\|_2^2 - \|c_0\|_2^2$ gives Equation (11). Because C has full column rank, this reconstruction is unique, proving that $z \mapsto q(z)$ is injective.

For an equivalent realization, metric transport gives $d'_T(\psi(z), c'_j) = d_T(z, c_j)$ for every j. The normalized exponential targets are therefore identical, $q'_j(\psi(z)) = q_j(z)$; invertibility of ψ transfers injectivity to the transformed coordinates.

For the loss decomposition, write $p_j^{\text{full}} = M_{\mathcal{Y}} \bar{p}_j$. Since $\sum_j q_j = 1$,

$$\begin{aligned} -\sum_j q_j \log p_j^{\text{full}} &= -\sum_j q_j \log \bar{p}_j - \log M_{\mathcal{Y}} \\ &= H(q) + \mathrm{KL}(q\|\bar{p}) - \log M_{\mathcal{Y}}. \end{aligned}$$

Both terms after $H(q)$ are nonnegative. If their sum is at most ϵ, each is at most ϵ, giving $M_{\mathcal{Y}} \geq e^{-\epsilon}$; Pinsker's inequality gives $\|\bar{p} - q\|_1 \leq \sqrt{2\epsilon}$. □

The same calculation gives a quantitative conditioning statement. Applying Equation (11) to any positive reference-token distribution r yields

$$\|\widehat{z}(r) - z\|_2 \leq \frac{\tau}{2} \|C^\dagger\|_2 \left\| \log \frac{r_{1:J}}{r_0} - \log \frac{q_{1:J}(z)}{q_0(z)} \right\|_2. \tag{14}$$

Thus reference configurations with better-conditioned spanning directions make coordinates more stable to output-log-odds error.

A.3 Approximation capacity of the SiLU interface

The preceding theorem establishes that the output objective contains enough information to distinguish continuous points in any equivalent reference realization supplying its targets. We now prove that the SiLU function class can approximate continuous maps carrying those coordinates between task and hidden-state spaces. The following is the formal version of Theorem 1.

Theorem 1 (formal). *Let $\mathcal{M} \subseteq \mathbb{R}^d$ and $\mathcal{Z} \subseteq \mathbb{R}^m$ be compact, and let $E^\star : \mathcal{M} \to \mathcal{Z}$ be continuous. Fix $\epsilon_E > 0$ and a compact $\mathcal{K}_E$ containing the closed ϵ_E-neighborhood of $\mathcal{Z}$. Suppose $\omega_T : \mathcal{K}_E \to \mathcal{K}_D$ is L_ω-Lipschitz for compact $\mathcal{K}_D$, and $D^\star : \mathcal{K}_D \to \mathbb{R}^d$ is L_D-Lipschitz. Then, for every $\epsilon_D > 0$, finite-width two-hidden-layer SiLU MLPs E_θ, D_ϕ satisfy*

$$\sup_{h\in\mathcal{M}} \|E_\theta(h) - E^\star(h)\|_2 < \epsilon_E, \qquad \sup_{z\in\mathcal{K}_D} \|D_\phi(z) - D^\star(z)\|_2 < \epsilon_D. \tag{15}$$

If the full-vocabulary output map Π of the remaining layers is L_Π-Lipschitz on the relevant states, then

$$\|\Pi(D_\phi(\omega_T(E_\theta(h)))) - \Pi(D^\star(\omega_T(E^\star(h))))\|_1 \le L_\Pi(\epsilon_D + L_D L_\omega \epsilon_E). \tag{16}$$

This theorem shows that a sufficiently wide steerer can approximate a continuous cross-space interface closely enough for its output after a task-space operation to approach the ideal interface's output. Suppose continuous ideal maps $E^\star$ and $D^\star$ implement the desired correspondence: $E^\star$ sends every task-bearing state h in the compact set $\mathcal{M}$ to the reference coordinate domain $\mathcal{Z}$, and $D^\star$ returns the operated coordinates to hidden-state space. Compactness restricts the claim to a bounded relevant region on which one network can attain a uniform worst-case error. Equation equation 15 says that sufficiently wide SiLU networks can uniformly approximate these maps, keeping $E_\theta(h)$ within ϵ_E of $E^\star(h)$ for every such state and $D_\phi(z)$ within ϵ_D of $D^\star(z)$ wherever decoding is required. The neighborhood $\mathcal{K}_E$ ensures that the approximate encoder output remains a valid input to the task operation ω_T, while $\mathcal{K}_D$ covers the resulting operated coordinates on which the decoder guarantee must hold.

The output bound in Equation (16) follows the same forward path. An encoder error of at most ϵ_E becomes at most $L_\omega\epsilon_E$ after ω_T and $L_D L_\omega \epsilon_E$ after the ideal decoder; replacing that decoder with D_ϕ adds at most ϵ_D. Processing the two hidden states through the remaining-layer output map Π therefore yields full-vocabulary distributions whose ℓ_1 distance is at most $L_\Pi(\epsilon_D + L_D L_\omega \epsilon_E)$. Thus the theorem guarantees that the SiLU interface has enough capacity to approximate the output of an ideal cross-space task operation. Equivalent reference realizations induce their corresponding ideal maps, so the same approximation argument applies without selecting one distinguished coordinate chart.

Proof of Theorem 1. SiLU is continuous and non-polynomial, so universal approximation applies coordinate-wise to continuous functions on compact subsets of Euclidean space (Leshno et al., 1993). It gives uniform approximations to $E^\star$ on $\mathcal{M}$ and $D^\star$ on $\mathcal{K}_D$. A second SiLU hidden layer can uniformly approximate the identity on the bounded first-layer image; splitting the error budget and padding unused units gives finite-width two-hidden-layer networks of the steerer form. Because $E^\star(\mathcal{M})$ lies in $\mathcal{Z}$, the neighborhood condition and the encoder bound ensure $E_\theta(\mathcal{M}) \subseteq \mathcal{K}_E$.

For the output bound, insert the ideal decoder at the learned coordinate:

$$\begin{aligned}
&\|D_\phi(\omega_T(E_\theta(h))) - D^\star(\omega_T(E^\star(h)))\|_2 \\
&\quad\le \|D_\phi(\omega_T(E_\theta(h))) - D^\star(\omega_T(E_\theta(h)))\|_2 \\
&\qquad + \|D^\star(\omega_T(E_\theta(h))) - D^\star(\omega_T(E^\star(h)))\|_2 \\
&\quad< \epsilon_D + L_D L_\omega \epsilon_E.
\end{aligned}$$

Applying the L_Π-Lipschitz output map proves Equation (16). □

A.4 Approximate composition through the coordinate interface

The first result shows that the training signal distinguishes continuous task coordinates, and the approximation theorem shows that the SiLU function class can realize continuous maps carrying

them between the two spaces. We now ask when the resulting interface also preserves the task's reusable operations. Let a group G act on $\mathcal{Z}$ through isometries ρ_g, so $\rho_g\rho_h = \rho_{gh}$. This includes day advances in Weekday and circular digit or residue advances in Number and Modular. The induced hidden-state operation is

$$\widehat{T}_g = D \circ \rho_g \circ E. \tag{17}$$

Although the coordinate actions compose exactly, repeatedly applying $\widehat{T}_g$ inserts a decoder–encoder round trip between successive task operations. Approximation error in this round trip can therefore break the group law after mapping the action back to hidden-state space. The following result quantifies how the two interface round-trip errors control both the difference between sequential and composed actions and return after a complete finite cycle.

Proposition 1 (Round-trip accuracy controls composition)**.** *Suppose D is L_D-Lipschitz and the interface satisfies*

$$\sup_{z\in\mathcal{Z}} \|E(D(z)) - z\|_2 \le \delta, \qquad \sup_{x\in\mathcal{M}} \|D(E(x)) - x\|_2 \le \epsilon, \tag{18}$$

with all iterates remaining in the stated domains. Then, for every $g, h \in G$ and $x \in \mathcal{M}$,

$$\|\widehat{T}_g(\widehat{T}_h(x)) - \widehat{T}_{gh}(x)\|_2 \le L_D\delta. \tag{19}$$

Moreover, if g has finite order K, then

$$\|\widehat{T}_g^K(x) - x\|_2 \le L_D(K-1)\delta + \epsilon. \tag{20}$$

This proposition shows that accurate round trips between coordinate and hidden-state spaces allow one shared transformation to approximately preserve task-space composition and finite-cycle return. Applying $\widehat{T}_h$ and then $\widehat{T}_g$ introduces the intermediate round trip $E(D(z))$, whose deviation from z is bounded by δ in Equation (18). Because ρ_g is an isometry and D is L_D-Lipschitz, this deviation produces at most $L_D\delta$ hidden-state error relative to the single composed operation $\widehat{T}_{gh}$ in Equation (19). Thus one shared coordinate-space operation inherits the task's group law approximately, unlike independently fitted source–target paths.

When g has finite order K, its ideal task-space action returns after K applications. The implemented action contains $K-1$ intermediate decoder–encoder round trips, contributing at most $L_D(K-1)\delta$, followed by the hidden-state reconstruction error $\|D(E(x)) - x\|_2 \le \epsilon$. This gives the cycle bound $L_D(K-1)\delta + \epsilon$ in Equation (20). For the seven-element cycles studied here, $K = 7$; the dense-decoding, one-step, and cycle-return experiments test whether the learned interface makes these errors small enough to preserve readable local movement and global return in practice.

Proof. Insert Equation (17) twice and compare with the task-space composition:

$$\begin{aligned}&\|D(\rho_g(E(D(\rho_h(E(x)))))) - D(\rho_g(\rho_h(E(x))))\|_2 \\ &\quad\le L_D\|E(D(\rho_h(E(x)))) - \rho_h(E(x))\|_2 \le L_D\delta,\end{aligned}$$

where the first inequality uses that ρ_g is an isometry. This proves Equation (19).

For repeated application, set $z_0 = E(x)$ and $z_{k+1} = E(D(\rho_g(z_k)))$. The first inequality in Equation (18) gives $\|z_{k+1} - \rho_g(z_k)\|_2 \le \delta$. Induction and isometry of ρ_g then yield $\|z_k - \rho_g^k(z_0)\|_2 \le k\delta$. Since $\widehat{T}_g^K(x) = D(\rho_g(z_{K-1}))$ and $\rho_g^K = I$,

$$\begin{aligned}\|\widehat{T}_g^K(x) - x\|_2 &\le \|D(\rho_g(z_{K-1})) - D(z_0)\|_2 + \|D(E(x)) - x\|_2 \\ &\le L_D(K-1)\delta + \epsilon.\end{aligned}$$

□

A.5 Dimensional requirements of a lossless interface

The approximation theorem assumes a coordinate space capable of carrying the task structure. We now ask how small that space can be while still permitting a continuous, lossless interface. This question directly motivates the controls in Table 4 and Section E.4, which reduce the RGB dimension or replace the Number circle by a line. The following result gives the corresponding topological lower bounds.

Proposition 2 (Dimension of a lossless continuous bottleneck)**.** *Let $\mathcal{Z}$ be compact. If continuous maps $A : \mathcal{Z} \to \mathbb{R}^r$ and $B : \mathbb{R}^r \to \mathcal{Z}$ satisfy $B \circ A = I_{\mathcal{Z}}$, then A is a topological embedding. Consequently, if $\mathcal{Z}$ contains a three-dimensional region with nonempty interior, then $r \geq 3$; if $\mathcal{Z}$ is homeomorphic to S^1, then $r \geq 2$.*

This proposition shows that a lossless continuous interface needs enough coordinate dimensions to preserve the topology of its task domain. If A encodes a task point in the compact reference domain $\mathcal{Z}$ into $\mathbb{R}^r$ and $B(A(z)) = z$ for every $z \in \mathcal{Z}$, then two distinct task points cannot share an encoding; continuity and compactness therefore make A an embedding that preserves the topology of $\mathcal{Z}$. When $\mathcal{Z}$ contains a three-dimensional region, no such embedding exists for $r < 3$, so a globally lossless RGB interface requires at least three coordinates. When $\mathcal{Z} \cong S^1$, embedding it into a one-dimensional line is impossible, so a lossless interface for the continuous circular realizations used by Weekday, Number, and Modular requires at least two coordinates. Accordingly, the reduced-dimensional RGB controls and linear Number control are structural collapses rather than equivalent reparameterizations. These are necessary conditions for exact continuous reconstruction; the experiments measure how the corresponding finite trained interfaces degrade in practice.

Proof. The identity $B \circ A = I_{\mathcal{Z}}$ makes A injective. A continuous injection from compact $\mathcal{Z}$ into the Hausdorff space $\mathbb{R}^r$ is a homeomorphism onto its image. Because $\mathcal{Z}$ contains a region with nonempty interior in $\mathbb{R}^3$, its topological dimension is at least three. Homeomorphism preserves this dimension, whereas every subspace of $\mathbb{R}^r$ has topological dimension at most r. Hence $A(\mathcal{Z}) \subseteq \mathbb{R}^r$ requires $r \geq 3$. If S^1 embedded in $\mathbb{R}$, its compact, connected image would be a closed interval. Removing an interior point disconnects such an interval, whereas removing any point from S^1 leaves a connected open arc, a contradiction. □

B Implementation and Evaluation Details

This appendix follows the methodological progression in the main text. We first specify the shared model, intervention, and output-anchor conventions, then give the architectures and optimization used for directly supervised Weekday and end-to-end Number and RGB steering. We finally specify Modular-specific optimization, its baselines, and checkpoint selection.

B.1 Language models and intervention layers

We evaluate base (non-instruction-tuned) causal language models from the Llama 3 and Qwen 3 families. The standard intervention point is $\lfloor 0.75L \rfloor$, where L is the number of Transformer layers. All task inputs are represented by the hidden state of the final non-padding prompt token at this layer. We do not pool over operand tokens or other prompt positions. The model depths and resulting intervention layers are listed in Table 7.

Llama3-70B and Qwen3-14B are loaded with four-bit NF4 quantization where required by memory constraints; other models are loaded in their native experiment precision. Frozen-model forward passes use the model dtype. The trainable circular encoder and decoder are optimized in float32 in the modular sweep, and decoded states are cast back to the frozen model dtype before insertion. Losses, distances, and evaluation accumulators are computed in float32.

B.2 Output anchors and remaining-layer evaluation

This section specifies how the output anchors in Equation (3) are implemented and evaluated. For each task element s, the LM-head row of its designated output token supplies the unembedding direction u_s; task-specific tokenization conventions are given in Section C.1. We use the resulting anchor to replace the final prompt-token state at layer ℓ, leave every other token position unchanged, and propagate the modified sequence through the remaining Transformer blocks, final normalization, and original LM head. An anchor is counted as readable when its designated token is the full-vocabulary top-1 prediction. We report this output-anchor accuracy separately from downstream aligner accuracy.

The base language model and LM head remain frozen throughout. Training updates only the encoder, decoder, or task-specific operand aligner.

B.3 Anchor scale and carrier sensitivity

We first specify how the scale α in Equation (3) is chosen. For every task, we run the unmodified carrier prompt p, extract its final-token state $h_{L,p}$ after the final normalization, normalize the target token's unembedding as $\bar{u}_s = u_s/\|u_s\|_2$, and set $\alpha_p = 2\|h_{L,p}\|_2$. The candidate anchor is then formed by Equation (3), propagated through the remaining layers, and its readability is evaluated by checking whether s is the full-vocabulary top prediction. As the logit expression in Section 3.2 makes explicit, this choice gives the targeted direction a norm larger than that of the original final state, helping its contribution to the target-token logit outweigh the carrier state and competing unembedding directions. The top-1 check remains necessary because the remaining layers are nonlinear. We use this same $2\|h_{L,p}\|_2$ scale for Weekday, Number, RGB, and Modular; cases requiring larger model-specific scales are examined in Section D.3.

We next test the corresponding claim from Section 3.2: once the anchor is sufficiently strong, its target need not be implied by the carrier prompt. The first control fixes `Today is` as the carrier, samples 512 distinct non-special vocabulary tokens as targets, and measures the fraction for which the resulting anchor makes the sampled token the model's full-vocabulary top prediction. The second fixes the seven weekday targets but replaces the carrier with 50 random-token sequences of the same length. For each sequence, we recompute the carrier-dependent scale, construct all seven anchors, and average their full-vocabulary top-1 accuracy across the resulting 350 prompt–target combinations. Table 8 reports both controls. The Llama models are nearly perfect under either randomization, and Qwen3-4B/8B remain strong; the weaker Qwen3-1.7B/14B results show that the fixed scale is not sufficient for every model. Overall, the controls support the mechanism in Section 3.2: anchor targets can be largely independent of prompt semantics when the calibrated direction dominates, but readability must still be verified and the scale occasionally increased.

B.4 Steerer architectures

The directly supervised Weekday steerer uses bias-enabled multilayer perceptrons

$$E : d \to 256 \to 256 \to 2, \tag{21}$$
$$D : 2 \to 256 \to 256 \to d, \tag{22}$$

with SiLU nonlinearities. It is trained in float32 and then cast to the model dtype for evaluation.

The standard Number and Modular circular steerers use bias-enabled networks

$$E : d \to 256 \to 256 \to 2, \tag{23}$$
$$D : 2 \to 256 \to 256 \to d. \tag{24}$$

Their shallow ablation removes the middle $256 \to 256$ layer from both sides. This is an architectural-capacity ablation, not a change in task-space dimension or training objective; results are reported in Section J.

The standard RGB steerer uses the corresponding bias-free three-dimensional architecture

$$E : d \to 256 \to 256 \to 3, \tag{25}$$
$$D : 3 \to 256 \to 256 \to d. \tag{26}$$

RGB encoder coordinates are bounded to the cube by the implementation's coordinate transform. This fixes the coordinate domain but does not directly supervise a pointwise correspondence with RGB coordinates.

B.5 Task-specific optimization

The default task-specific optimization settings are summarized in Table 9.

Number, RGB, and Modular use temperature $\tau = 0.1$ and gradient-norm clipping at 1. Their coordinate, reconstruction, and explicit distance-correlation weights are zero in the standard setting. The Weekday construction uses 1,024 points per arc and repeats anchors 100 times.

The soft-target cross-entropy uses full-vocabulary probabilities, so it both matches relative geometry among the reference outputs and rewards probability mass assigned to that set.

The Modular steerer is exposed to 18,000 pair-disjoint training prompts as carrier contexts. To limit memory, contexts are extracted in slices of 256; each optimizer microbatch uses 16 contexts. Checkpoint comparisons accumulate 128 examples. Slicing changes storage and extraction order, not the training pool.

B.6 Baselines

The modular comparison uses the same 18,000 training prompts and 500 test prompts for all methods. Logistic regression first standardizes the extracted last-token state as $\widetilde{h} = (h - \mu) \oslash \sigma$, then fits

$$p_{\log}(r \mid h) = \mathrm{softmax}(W\widetilde{h} + b)_r \tag{27}$$

by multinomial cross-entropy with L_2 regularization ($C = 1$) and at most 1,000 optimization iterations.

LoRA and Prefix instead minimize the full-vocabulary next-token objective $\mathcal{L}_{\mathrm{PEFT}} = -\sum_n \log p(y_n \mid p_n)$. Each canonical adapter is trained only at the model-specific intervention layer in Table 7: layers 24, 24, 21, 60, 21, 27, 27, and 30 for Llama3-8B, Llama3.1-8B, Llama3.2-3B, Llama3-70B, Qwen3-1.7B, Qwen3-4B, Qwen3-8B, and Qwen3-14B, respectively. LoRA modifies only the q, k, v, and output attention projections and the gate, up, and down MLP projections in transformer block ℓ:

$$W' = W + \frac{\alpha}{r} BA, \qquad A \in \mathbb{R}^{r \times d_{\mathrm{in}}}, \quad B \in \mathbb{R}^{d_{\mathrm{out}} \times r}, \tag{28}$$

with only A and B trainable. Direct Prefix tuning instead prepends trainable key–value rows only at block ℓ,

$$\mathrm{Attn}_\ell(Q, K, V) \longmapsto \mathrm{Attn}_\ell\big(Q, [K_{\mathrm{pre}}; K], [V_{\mathrm{pre}}; V]\big), \tag{29}$$

and trains K_{pre} and V_{pre} while leaving all model weights fixed. For each model, we choose the smallest integer LoRA rank or prefix length whose trainable count reaches $840d$; LoRA uses $\alpha \approx r/2$ and dropout 0.05. Exact configurations are reported in Section G.1. Both methods use one epoch, learning rate 2×10^{-4}, weight decay 10^{-2}, and effective batch size 32; memory-specific microbatching does not change either parameterization. The Modular aligner is trained on the same schedule—one epoch over the same 18,000 optimization prompts, effective batch size 32, and weight decay 10^{-2}—with learning rate 5×10^{-4} for its randomly initialized branches. For the Qwen3-4B layer sweep, a separate LoRA or Prefix adapter is retrained at each evaluated layer rather than sharing parameters across layers.

The comparison to nonlinear steering methods follows the Manifold Steering paper setting on Llama3.1-8B at layer 28 with its 49-prompt weekday corpus, its PCA-64 setting (Section H), centroid spline $\gamma_{s\to t}(u)$, and carrier-residual preservation: a source-state residual is added to $\gamma_{s\to t}(u)$ along the fitted path. We use shared source and target states and report target correctness separately from path and repeated-composition metrics. Implementation provenance and metric definitions are detailed in Section H.

B.7 Checkpoint selection

For Weekday, we retain the steerer checkpoint with the lowest directly supervised encoder–decoder objective on the complete fixed set of circular interpolations. Number and RGB instead select the checkpoint with the highest full-vocabulary nearest-reference accuracy on the current sampled training batch, breaking ties by lower total training loss. The Modular steerer uses the same rule but accumulates the criterion over 128 prompt-conditioned training examples before each comparison. After freezing this steerer, we train the Modular aligner for one epoch and evaluate its final iterate; no validation or test example is used for checkpoint selection or gradient updates.

C Tasks, Prompts, and Modular Training

We present the task-specific inputs in the same order as the main experiments: the directly supervised Weekday cycle, the sparsely supported Number cycle, the three-dimensional RGB domain, and the prompt-conditioned Modular application. Shared tokenization and intervention-position conventions come first; the final subsections give the Modular data split, contextual steerer training, operand aligner, and exact composition rule.

C.1 TOKENIZATION CONVENTION

Every reported task element is paired with a designated output token. The Weekday, Number, and Modular labels used in our experiments each tokenize to a single token. For RGB, if a color name tokenizes into multiple tokens, we use its first token as the output label. Interventions and extracted features use the last non-padding token of the complete query prompt. Thus the method does not depend on locating individual operand or color-word tokens inside a prompt.

C.2 WEEKDAY TASK

The directly supervised Weekday experiment uses the carrier prompt

```
Today is
```

and the ordered targets

```
Monday, Tuesday, Wednesday, Thursday, Friday,
Saturday, Sunday.
```

Their order defines the seven-cycle in the Weekday reference realization. This carrier is used to study the steering operation in isolation: because the prompt does not specify any particular weekday as its answer, successful steering supports the claim in Section 3.2 that an output anchor can make its designated token readable independently of the carrier's original semantics.

C.3 NUMBER TASK

Number uses the fixed carrier prompt

```
The answer is
```

and labels `0`,...,`6` arranged in cyclic order in the Number reference realization. The support elements are `0`, `3`, and `6`; their output anchors define the closed piecewise-linear path $a_0 \to a_3 \to a_6 \to a_0$ from which the circular interface is trained. Continuous phases ϕ are sampled over one full period and mapped to

$$c(\phi) = \left(\cos\frac{2\pi\phi}{7}, \sin\frac{2\pi\phi}{7}\right). \tag{30}$$

For each phase, we instantiate the coefficient rule in Equation (4) by linear interpolation on the corresponding segment:

$$h_{\text{mix}}(c(\phi)) = \begin{cases} \left(1-\dfrac{\phi}{3}\right)a_0 + \dfrac{\phi}{3}a_3, & 0 \le \phi < 3, \\ \left(1-\dfrac{\phi-3}{3}\right)a_3 + \dfrac{\phi-3}{3}a_6, & 3 \le \phi < 6, \\ (1-(\phi-6))\,a_6 + (\phi-6)a_0, & 6 \le \phi < 7. \end{cases} \tag{31}$$

The first two segments span three phase units, whereas the final segment spans one and connects a_6 back to a_0, making the hidden-state path continuous across the prescribed 6–0 adjacency. The complete task-element assignment is shown in Table 10.

C.4 RGB TASK

RGB uses the fixed carrier prompt

```
This color is
```

and three basis anchors corresponding to `red`, `green`, and `blue`. Its reference realization is the RGB cube $[0,1]^3$; training samples coordinates uniformly from this cube and constructs the associated anchor mixture. The pool of 64 task elements includes cube corners, primary and secondary colors, neutral colors, edge and face representatives, and additional named colors throughout the cube; their names and coordinates are listed in Table 11.

C.5 MODULAR ARITHMETIC PROMPTS

The standard benchmark contains ordered operand pairs $(a, b) \in \{0, \ldots, 29\}^2$ and modulus seven. The canonical zero-shot template is

```
{a}+{b} mod 7, the answer is
```

with target $(a + b) \bmod 7$.

C.6 PAIR-DISJOINT MODULAR SPLIT AND CONTEXTUAL TRAINING

We split unique ordered operand pairs before drawing repeated prompt examples. Pairs are stratified by the result $(a + b) \bmod k$. With test fraction 0.2, all members of the swap orbit

$$[(a, b)] = \{(a, b), (b, a)\} \tag{32}$$

are assigned to the same side of the split. This prevents a model trained on (a, b) from being tested on the semantically identical commuted pair (b, a). The remaining non-test pair pool is split again, with validation fraction 0.1, using the same result stratification and swap-orbit constraint. We verify zero pair overlap and require the optimization split to cover every absolute operand value in both roles.

The canonical split contains 645 optimization, 71 validation, and 184 test ordered pairs. Within each split, we repeatedly shuffle and cycle through its pair pool to form 18,000 optimization examples, 500 validation examples, and 500 test examples; pair repetition is therefore permitted, but pair identity never crosses splits. Only the optimization examples are used for training. The validation pairs are held out from the steerer, the aligner, and every baseline, and the reported protocol uses them for no purpose: neither the steerer's checkpoint rule, which accumulates its criterion over training examples (Section B.7), nor the aligner, which is trained for one epoch and evaluated at its final iterate, consults them. The main text therefore describes the split as training and test only.

The Modular steerer uses the same full-vocabulary output objective as in Equation (8), but its carrier is prompt-conditioned rather than fixed. For each optimization instance, we extract the layer-ℓ context of its modular-arithmetic prompt, replace the final-token state with a sampled residue-anchor mixture, and backpropagate through the remaining layers to update the encoder and decoder. Training across these contexts teaches the decoder to make each of the seven residue labels the model's top prediction across the varying operand expressions on which the downstream system will operate. We freeze the fitted steerer before training the operand aligner.

C.7 MODULAR OPERAND ALIGNER AND EXACT COMPOSITION

After freezing the Modular steerer, we train an aligner to determine which residue coordinate its decoder should receive for each arithmetic prompt. The aligner contains two independent role-specific networks, which we call branch A and branch B. For the prompt $p(a, b)$ and its extracted last-token state $h_{a,b}$, branch A predicts the first operand a, whereas branch B predicts the second operand b. Each branch maps

$$d \to 128 \xrightarrow{\text{SiLU}} \text{Dropout}(0.2) \to 30, \tag{33}$$

and the two branches do not share parameters. Let $\ell_A = f_A(h_{a,b}), \ell_B = f_B(h_{a,b}) \in \mathbb{R}^{30}$ be their logits. Then $P_A = \text{softmax}(\ell_A)$ and $P_B = \text{softmax}(\ell_B)$ are distributions over the possible absolute values $0, \ldots, 29$ of the first and second operand, respectively.

To compute the modular sum, we first fold each absolute-value distribution into a distribution over residues modulo k, then sum the probabilities of all residue pairs whose modular sum is r:

$$\begin{gathered} \overline{P}_A(i) = \sum_{u:\, u \bmod k = i} P_A(u), \qquad \overline{P}_B(j) = \sum_{v:\, v \bmod k = j} P_B(v), \\ P_R(r) = \sum_{i=0}^{k-1} \overline{P}_A(i)\overline{P}_B((r - i) \bmod k). \end{gathered} \tag{34}$$

Thus each summand in $P_R(r)$ represents one possible pair of operand residues whose sum is r modulo k. The aligner objective supervises the two absolute operands:

$$\mathcal{L}_{\text{align}} = -\log P_A(a) - \log P_B(b). \tag{35}$$

Each optimization example is used in its original order; we neither add the reversed prompt as data augmentation nor average predictions from both prompt orders at inference. We apply softmax directly to $f_A(h_{a,b})$ and $f_B(h_{a,b})$, fold the resulting distributions modulo k, and combine them using Equation (34). We then take $\widehat{r} = \arg\max_r P_R(r)$. Let $z_j = E_\theta(a_j)/\|E_\theta(a_j)\|_2$ be the learned coordinate of residue anchor a_j. The decoder produces $h_{\text{out}} = D_\phi(z_{\widehat{r}})$, which replaces the final prompt-token state at layer ℓ; running the remaining layers from this state produces the output token.

The circular convolution in Equation (34) is prescribed rather than learned and cannot amplify operand-distribution error beyond the sum of the two input errors. Specifically, let p_A, p_B be the ideal residue distributions—point masses at $a \bmod k$ and $b \bmod k$—and let $\widehat{p}_A, \widehat{p}_B$ be the aligner's folded residue distributions. With $*_k$ denoting circular convolution,

$$\|\widehat{p}_A *_k \widehat{p}_B - p_A *_k p_B\|_1 \le \|\widehat{p}_A - p_A\|_1 + \|\widehat{p}_B - p_B\|_1. \tag{36}$$

Indeed,

$$\widehat{p}_A *_k \widehat{p}_B - p_A *_k p_B = (\widehat{p}_A - p_A) *_k \widehat{p}_B + p_A *_k (\widehat{p}_B - p_B), \tag{37}$$

and Young's inequality together with the unit L_1 norm of probability distributions gives the bound. The exact convolution therefore introduces no additional learned computation error: a wrong final token can arise because the aligner predicts inaccurate operand distributions, or because the decoder and remaining layers do not make the correctly computed residue the model's top prediction.

We train the aligner for one epoch with AdamW, batch size 32, learning rate 5×10^{-4}, weight decay 10^{-2}, and gradient clipping at 1. The final iterate is evaluated directly, without validation-based checkpoint selection.

D Supplementary Cyclic-Steering Results

The main-text Weekday experiment first examines whether the steerer organizes the seven output anchors into the prescribed cycle and then tests whether one shared rotation remains valid under continuous and repeated composition. This appendix follows the same progression across the remaining models: it examines the learned coordinates, replicates the continuous comparison with vector steering, diagnoses failures caused by unreadable anchors, and finally tests the output-supervised training variant.

D.1 Replication of the Weekday Encoding in Figure 2

Because the main-text visualization uses Qwen3-4B, we first test whether the same directly supervised interface can realize the prescribed Weekday cycle across model families. We train each model with the same coordinate and reconstruction losses and show the resulting anchor encodings in Figures 10 to 16. As in Figure 9, the learned coordinates organize the seven output anchors around the specified circle, but the resulting outputs are not uniformly correct across models. Five of the seven additional models decode all seven positions correctly, whereas Qwen3-1.7B decodes none and Qwen3-14B decodes five. Thus direct supervision succeeds across most models but also exhibits failure cases, whose source we examine further in Section D.3.

D.2 Replication of Figure 3 for Other Models

We next test whether the advantage over vector steering in Figure 3 persists across the remaining models. For each model, we start from Monday and compose 0.01-day steps over one complete cycle. The Weekday steerer repeatedly applies the same $2\pi/700$ coordinate rotation, whereas vector steering repeatedly adds the same fraction of the Tuesday-minus-Monday displacement. Figures 17 to 23 shows the clearest full-cycle replication for Llama3-8B, Llama3.1-8B, Llama3-70B, and Qwen3-8B, while vector addition continues along an open path. The remaining models expose stricter failures under fractional composition: Llama3.2-3B loses the intended phase near the latter part of the cycle, Qwen3-1.7B remains dominated by one weekday, and Qwen3-14B does not complete the return to Monday. The two Qwen failures are already present at the discrete outputs in Section D.1, so we next test whether their anchors are sufficiently readable before attributing the failures to the learned cyclic operation.

D.3 Anchor strength explains two cross-model failures

The path replications expose failures for Qwen3-1.7B and Qwen3-14B, but an incorrect output does not by itself show whether the learned cyclic operation failed or whether its input anchors were never readable. To separate these possibilities, we inspect the anchors before training the steerer. At the default scale $\alpha = 2\|h_{L,p}\|_2$, inspection of the corresponding output anchors finds that the failed states predict background vocabulary items—most often digit tokens—rather than another English weekday. This suggests that the injected weekday direction is too weak to overcome competing background logits. We therefore sweep α while leaving the carrier prompt, layer, and unembedding direction fixed; this diagnostic retrains no steerer and isolates only the readability of the constructed anchors. As Figure 24 shows, Qwen3-1.7B improves from $0/7$ readable anchors at the default scale to $7/7$ at multiplier 16, while Qwen3-14B improves from $1/7$ to $7/7$ at multiplier 64. This intervention-strength dependence is consistent with the anchor logit in Section 3.2: increasing α strengthens the target-token component until it overtakes the background vocabulary. General scale selection and carrier sensitivity are described in Section B.3.

Table 6: Increasing the Qwen3-1.7B weekday-anchor scale restores full-vocabulary top-1 accuracy for the fixed weekday carrier and substantially improves 512 random target tokens and weekdays under 50 random carrier prompts.

Anchor scale	Weekday	Random targets	Random carriers
$2\times$ (default)	0.000	0.000	0.146
$128\times$	1.000	0.965	1.000

To determine whether improved readability also restores the downstream cyclic operation, we rerun Qwen3-1.7B at the conservative scale $\alpha = 128\|h_{L,p}\|_2$, changing no other model, layer, carrier, architecture, or optimization setting. In Figure 25, both reconstructed-anchor and repeated one-day accuracy rise from $0/7$ to $7/7$, and the exact seven-step return changes from incorrect to correct. The direct intervention diagnostic in Table 6 likewise improves from 0% to 100% on the seven weekdays, from 0% to 96.5% on 512 random target tokens, and from 14.6% to 100% across random carrier prompts. Thus the earlier failure was not evidence that the model could not support a circular interface: increasing α constructs states whose target-token components reliably dominate the background vocabulary.

The stronger anchors also make the recursive path substantially more informative. Figure 26 now traverses the weekday logits in their cyclic order instead of remaining nearly constant. To quantify this change, at each sampled point on the circle we select the weekday with the largest logit and score whether it corresponds to the nearest weekday phase; this accuracy rises from 14.4% to 55.6%. The gain therefore extends beyond the seven anchor endpoints to repeated small-step transformations. Together, these results show that increasing α can rescue an otherwise completely failed model by constructing more effective token-bearing states for the steerer.

D.4 Output-supervised weekday ablation

The Weekday steerer used in the main text and the preceding experiments receives the intended circular coordinates directly through its encoder loss. To test whether a compatible circular realization can instead emerge without internal coordinate or reconstruction targets, as in Number and RGB, we train a Qwen3-4B Weekday steerer using only the end-to-end output objective in Equation (8).

As Table 13 shows, the resulting steerer reaches 96.5% accuracy on 512 randomly sampled phases and decodes all seven weekday phases correctly, compared with 41.4% for the direct hidden-state mixture and 28.9% for a linear probe. The left panel of Figure 27 shows that its encoder places the seven hidden-state mixtures in Weekday order around the circular coordinate domain. At these seven exact phases, the complete steerer decodes all labels correctly, whereas the direct hidden-state mixture succeeds for only three (right). Thus the Weekday cycle need not be imposed through direct coordinate regression: output-level training can jointly learn an internal realization of that structure and make it readable by the model.

E SUPPLEMENTARY NUMBER RESULTS

The main-text Number experiments show that end-to-end output supervision can induce a circular realization from only three support anchors. This section specifies the linear-regression baseline and visualization alignment, then reports cross-model replications, exact wrong-geometry controls, and diagnostics that separate dense decoding from one-step and closure behavior.

E.1 LINEAR-REGRESSION BASELINE

The linear-regression baseline tests whether direct coordinate supervision and a single affine map suffice to recover the Number circle from the same hidden-state mixtures. We independently sample $N = 512$ phases, construct $z_n = c(\phi_n)$ and $h_n = h_{\mathrm{mix}}(z_n)$ using the three-anchor instantiation of Equation (4), and fit $f_{W,b}(h) = Wh + b \in \mathbb{R}^2$ by

$$(W^*, b^*) = \operatorname*{argmin}_{W,b} \sum_{n=1}^{N} \|Wh_n + b - z_n\|_2^2 + \lambda\big(\|W\|_F^2 + \|b\|_2^2\big), \qquad \lambda = 10^{-4}. \tag{38}$$

We solve this ridge objective in closed form on a training set independent of the 512 evaluation phases. At evaluation, we normalize $f_{W^*,b^*}(h_{\mathrm{mix}}(z))$ to the unit circle and predict the digit whose reference coordinate is nearest in Euclidean distance. Only W and b are fitted; the language model remains unchanged. This provides the directly supervised comparator used in the cross-model analysis below.

E.2 ALIGNMENT OF EMERGENT NUMBER COORDINATES

The unit circle used for sampling is the Number reference realization. We test whether output supervision induces a compatible internal realization by aligning its global rotation and display scale with this reference circle. For sampled phase ϕ, we plot $\widetilde{z}(\phi) = \|h_{\mathrm{mix}}(c(\phi))\|_2 E_\theta(h_{\mathrm{mix}}(c(\phi)))$. If the first sampled point is (a, b), every point (x, y) is mapped by the same similarity transform

$$(x, y) \longmapsto \frac{(ax + by,\ ay - bx)}{a^2 + b^2},$$

which places that first phase at $(1, 0)$ and fixes the chart's displayed rotation and radius. No other anchor or interpolated point is used to fit the transform, and the learned orientation may still be globally reflected. Consequently, the circular shape, cyclic neighborhoods, and 6–0 adjacency visible after alignment support an emergent circular realization. This visual agreement does not by itself establish exact preservation of distances or cyclic operations, which we evaluate separately.

E.3 REPLICATION OF TABLE 2 AND FIGURE 4 FOR OTHER MODELS

The model-specific panels repeat the main behavioral and encoding analyses at each model's three-quarter-depth intervention layer, using the same seven phases and support anchors a_0, a_3, and a_6. The behavioral results in Table 14 show that the output-supervised interface reaches 87.9–97.9% dense accuracy across the eight models, compared with 18.9–42.8% for the direct hidden-state mixture and 12.9–29.9% for linear regression. The learned interface therefore exceeds the stronger baseline by at least 49.6 percentage points in every model, while all exact label anchors remain readable. Independent encoding plots in Figures 28 to 34 test whether these sparsely supported mixtures exhibit the task's phase order and 6–0 neighborhood, rather than merely separating the seven digit labels. Across models, the encoded mixtures form continuous closed trajectories with neighboring digit phases remaining nearby, although their precise shapes vary. These panels provide qualitative evidence of circular organization; the behavioral advantage above is established independently by the dense-accuracy comparison.

E.4 NUMBER WRONG-GEOMETRY PROTOCOL

To test whether the learned Number behavior depends on the intended circular relations, we construct two matched training controls. Let $c(j) = (\cos(2\pi j/7), \sin(2\pi j/7))$. The permuted-circle control uses

$$c_{\mathrm{perm}}(j) = c(\pi_j), \qquad (\pi_0, \ldots, \pi_6) = (0, 2, 5, 1, 6, 3, 4), \tag{39}$$

which retains seven equally spaced circle points but changes their semantic neighborhoods. The line control uses

$$c_{\text{line}}(j) = \big(2\sin(\pi/7)(j-3), 0\big), \tag{40}$$

so consecutive digits retain the circle's adjacent chord distance while the 6–0 adjacency and circular closure are removed. Section A.5 formalizes why this one-dimensional control cannot be a lossless continuous reparameterization of the circle. Between adjacent phases, each control linearly interpolates its corresponding vertices (and normalizes the permuted-circle interpolation to the unit circle). The hidden-state mixtures remain unchanged; only the coordinates, distances, and resulting $q(z)$ are replaced. We independently retrain each variant with the same architecture, optimizer, and checkpoint-selection rule as the standard circle. Dense accuracy evaluates 700 uniformly spaced phases by full-vocabulary top-1 agreement with the digit assigned to each semantic phase. Thus the comparison changes task relations while controlling the model-side inputs and training capacity.

E.5 Continuous decoding between Number elements

Dense accuracy summarizes performance over the complete Number circle, but does not show where the interface succeeds or fails between digit phases. We therefore localize decoding errors by sampling 720 phases ϕ uniformly over one period and passing each mixture through the complete interface, giving $h_\phi = D_\phi(E_\theta(h_{\text{mix}}(c(\phi))))$. After replacing the layer-ℓ carrier state with h_ϕ, we run the remaining layers and record the highest-logit digit. The direct-mixture baseline instead inserts $h_{\text{mix}}(c(\phi))$. In Figure 35 and Figures 36 to 42, the learned interface follows the reference bands with 87.9–96.7% candidate-digit accuracy across models, compared with 19.2–42.5% for direct mixtures. Its advantage therefore extends between the seven exact digit phases rather than only to the labeled locations.

E.6 Number wrong-geometry controls across models

The RGB controls in Table 4 show that replacing a task-consistent geometry during training degrades steering. We test the corresponding question for Number: whether its behavior likewise depends on the prescribed circular relations. We retrain matched interfaces with two wrong geometries while keeping the hidden-state mixtures, architecture, optimization, and task labels fixed. The permuted-circle control assigns the digits to circular positions in the order $0, 2, 5, 1, 6, 3, 4$, preserving a two-dimensional circle but changing which digits are neighbors. The line control instead places $0, \ldots, 6$ in order with adjacent spacing matched to the circle, preserving local linear order while removing the prescribed 6–0 adjacency and circular closure.

Results across all models appear in Table 15. The task-consistent circle attains 87.4–96.7% dense accuracy and is best for every model, compared with 64.7–74.4% for the permuted circle and 80.1–84.9% for the line. Thus changing digit neighborhoods causes the larger loss in dense decoding. The line produces a smaller accuracy loss because it retains the local order $0, \ldots, 6$, but it has the largest seven-step closure error for every model. Together, these controls show that local neighborhoods and global cyclic closure support distinct aspects of Number steering rather than arising from the interface architecture alone.

The aggregate results above do not reveal whether an incorrect geometry disrupts ordinary local movement, generalization across the $6 \to 0$ boundary, or repeated reuse of the same operation. To separate these effects, we encode the seven exact-phase mixtures as $z_j = E_\theta(h_{\text{mix}}(c(j)))$ and fit one affine step $T(z) = Az + b$ by minimizing $\sum_{j=0}^{5} \|T(z_j) - z_{j+1}\|_2^2$. The fit uses only the six ordinary transitions from $0 \to 1$ through $5 \to 6$; $6 \to 0$ therefore tests transfer across the unseen boundary. Let $\widehat{s}(z)$ be the full-vocabulary top-1 digit obtained after decoding z and running the remaining layers, let $z_\phi = E_\theta(h_{\text{mix}}(c(\phi)))$, and let $s^*(\phi)$ be the digit nearest phase ϕ. We measure dense-path accuracy $A_{\text{dense}} = \frac{1}{700}\sum_n \mathbf{1}[\widehat{s}(z_{\phi_n}) = s^*(\phi_n)]$, non-boundary step accuracy $A_{\text{step}} = \frac{1}{6}\sum_{j=0}^{5} \mathbf{1}[\widehat{s}(T(z_j)) = j+1]$, and the held-out boundary score $A_{6\to0} = \mathbf{1}[\widehat{s}(T(z_6)) = 0]$. Repeated composition is measured by the seven-step relative closure error

$$E_{\text{close}} = \frac{\frac{1}{7}\sum_{j=0}^{6} \|T^7(z_j) - z_j\|_2}{\frac{1}{7}\sum_{j=0}^{6} \|z_j\|_2}, \tag{41}$$

where lower is better. Applying these diagnostics to all three realizations gives the results in Figures 43 to 50. Under the task-consistent circle, all eight models correctly decode all six fitted transitions. The held-out $6 \to 0$ transition succeeds only for Llama3.1-8B, Qwen3-1.7B, and Qwen3-14B, so accurate local movement still does not guarantee wraparound generalization. Closure error ranges from 0.401 to 1.077 for the correct circle, 0.827 to 1.082 for the permuted circle, and 1.137 to 1.755 for the line. The line again attains perfect fitted-step accuracy in all eight models while failing every boundary test and having the largest closure error. The permuted circle always reduces dense accuracy but can have lower numerical closure error than the correct circle, because closure alone does not test whether intermediate states follow the intended digit order. Together, the diagnostics separate semantic ordering, boundary generalization, and repeated closure rather than treating any one quantity as a complete test of the cyclic action.

F Supplementary RGB Results

After the one-dimensional circular Weekday and Number settings, RGB evaluates whether output supervision induces an internal geometric realization consistent with bounded three-dimensional RGB structure. The main text reports Qwen3-4B; here we provide the corresponding quantitative results and model-specific visualizations for the remaining systems. Accuracy measures whether a decoded mixture produces the intended color token under the frozen remaining layers. Geometry correlation is complementary: it asks whether pairwise distances in the learned representation track distances in the declared RGB cube rather than merely separating a finite label set.

F.1 Linear Color Probe

The linear color probe tests whether direct coordinate supervision and a single affine map suffice to recover RGB coordinates from the frozen model's own color-name states. For each prompt "This color is *name*", we extract the layer-ℓ state $h_i \in \mathbb{R}^d$ of the final color-name token and pair it with the color coordinate $z_i \in [0,1]^3$. A fixed split assigns 32 colors to training and 32 to evaluation. We fit $f_{W,b}(h) = Wh + b$ by

$$(W^*, b^*) = \operatorname*{argmin}_{W,b} \sum_{i \in \mathcal{I}_{\text{train}}} \|Wh_i + b - z_i\|_2^2 + \lambda\big(\|W\|_F^2 + \|b\|_2^2\big), \qquad \lambda = 10^{-4}, \tag{42}$$

using the closed-form ridge solution. At evaluation, we compute $f_{W^*,b^*}(h_i)$ for each held-out color and predict the task element with the nearest reference coordinate. Only W and b are fitted; the language model remains unchanged. This is the linear color probe used in the cross-model analysis below.

F.2 Unembedding coordinates of color tokens

The linear color probe asks whether RGB geometry is linearly readable from the model's hidden states; this section asks the same of its unembedding. Let $\bar{u} = \frac{1}{V}\sum_v u_v$ be the mean unembedding row and $B = [u_{\text{red}} - \bar{u},\ u_{\text{green}} - \bar{u},\ u_{\text{blue}} - \bar{u}]$. For each of the 64 color tokens j we compute $\widetilde{z}_j = B^+(u_j - \bar{u}) \in \mathbb{R}^3$, which equals c_j exactly when the token's centered unembedding is the corresponding mixture of the red, green, and blue unembeddings. The three primaries sit at their corners by construction, so the test concerns the other 61 colors. In Figure 51, those colors collapse into one small cluster instead of filling the cube, whereas the decoded coordinates of the same colors spread across it. Their pairwise distances correlate with RGB distances at $r = 0.234$ on Qwen3-4B and at $r = 0.226$–0.237 across the eight models, with MSE 0.34–0.36 against the reference coordinates; the decoded coordinates reach $r = 0.707$ on Qwen3-4B. The readout therefore does not place intermediate color-token unembeddings at the corresponding RGB mixtures of the red, green, and blue token unembeddings. This mismatch helps explain why directly inserting the same linear mixtures of the three unembedding-derived anchor states does not reliably produce intermediate colors (Table 3).

F.3 Where the interface meets the model's vocabulary

Section F.2 asks whether the model's color-token unembeddings carry RGB geometry. This section asks the complementary question about the interface itself: which vocabulary directions does it read

from, and which does it write into? The interface touches the residual stream at exactly two places, the encoder's first layer and the decoder's last, and both are linear maps whose singular vectors are hidden-state directions. Nothing supervises those weights: the encoder never receives z, and training only matches the decoded state's output distribution to Equation (7).

Let $W_E \in \mathbb{R}^{256\times d}$ be the encoder's first layer and $W_D \in \mathbb{R}^{d\times 256}$ the decoder's last, with $d = 2560$ for Qwen3-4B. The left singular vectors of $W_E^\top$ and of W_D are orthonormal bases of the 256-dimensional subspaces the interface reads from and writes into. A singular vector is defined only up to sign, so for each of the 30 leading directions u_i of a side we decode the ten largest-logit tokens of both $W_U u_i$ and $-W_U u_i$, where W_U is the LM head. Each decoded token is then tested against a color lexicon: the 64 trained color names, a list of common English color words together with a few color-vocabulary terms such as `rgb` and `hue`, and a list of Chinese color words. A Latin token matches when its letters, once case, spacing, and punctuation are stripped, equal a lexicon entry, so that `/blue` and `␣BLUE` both count as blue; a Chinese token matches when it contains one, since Chinese compounds carry no separator and a single token can read *green development*. A direction counts as a color direction when either of its signs puts at least one matching token among its ten, and we report how many of the 30 do.

Two controls calibrate that count. The first decodes 2,000 random unit directions the same way and records how often one of them produces a color token at all. The second looks at each subspace as a whole instead of at its 30 leading directions. Let P be the orthogonal projection onto the subspace; for any vector v, the ratio $\|Pv\|^2/\|v\|^2 \in [0, 1]$ is the fraction of v that the subspace keeps, equal to 1 when v lies inside it and to 0 when v is orthogonal to it. We report this ratio for the 64 color tokens' unembedding rows and, as a baseline, for 2,000 rows drawn at random from the vocabulary. An isotropic direction spreads its norm evenly over the d axes, so a 256-dimensional subspace keeps $256/2560 = 0.100$ of it on average; unembedding rows are not isotropic, so we compare against the random rows rather than against this figure.

Table 17 lists every direction that decodes a color token. Six of the encoder's 30 do. Only 1.75% of the random directions produce one, so $30 \times 0.0175 = 0.5$ of the 30 would be expected by chance and six sits far out in that tail ($p = 1.2 \times 10^{-5}$, one-sided binomial). Five of the six decode a primary color name—blue at u_9 and $-u_{25}$, green at $-u_{14}$ and u_{23}, red at $-u_{26}$—and the sixth decodes the literal token `(rgb`; no other color name appears.

The subspace test agrees and identifies what is being kept. The read subspace keeps 0.9998, 0.9996, and 0.9997 of the three anchor states, so the encoder's inputs lie almost entirely inside it, but only 0.609 of the carrier state $h_{\ell,p}$ that those anchors share. Since each anchor is that carrier plus a scaled color-token unembedding (Equation (3)), keeping the anchors while discarding part of the carrier means keeping the unembedding part: the subspace retains 0.860, 0.861, and 0.860 of the bare red, green, and blue unembedding rows. The alignment stops at those three. The other 61 color names sit at 0.114 ± 0.028 and the 2,000 random vocabulary rows at 0.111 ± 0.019, both slightly above the 0.100 of an isotropic direction because unembedding rows share a common component; a one-sided Mann–Whitney U test of the 61 color values against the 2,000 random ones gives $p = 0.86$, that is, no evidence that a non-primary color token is retained any better than an arbitrary token. The decoder differs on both tests. None of its 30 leading write directions decodes a color token, and its write subspace retains 0.103 ± 0.004 of the primaries, 0.100 ± 0.009 of the other colors, and 0.096–0.107 of the carrier and the anchor states: chance everywhere.

Reconciling task coordinates with hidden states therefore leaves a visible trace in the encoder's weights. Trained only through output distributions, it internalizes the unembedding directions in which its inputs are expressed and adopts them as the coordinates it reads, and it does so legibly: its leading singular directions decode color names in two languages rather than any particular string. Retaining the anchor span is necessary for recovering z at all; what the measurements add is how specific that retention is. The subspace singles out the three unembedding rows its own inputs are built from, at 0.86, and leaves the model's other 61 color words at the level of an arbitrary token. The encoder is thus aligned with its own coordinate axes, not with the model's color vocabulary as a whole. The decoder acquires neither. Its write directions carry no readout content, so the states it emits are not scaled color-token unembeddings, and converting them into the correct color token is left to the layers above. This is consistent with Section F.6, where accuracy tracks the remaining layers' readability of the decoded state rather than the interface's own geometric error.

F.4 Alignment of emergent RGB coordinates

The RGB cube is the reference realization; learned coordinates may use a different origin and basis. We test their structural correspondence using a restricted affine alignment. For visualization only, let $u_i = E_\theta(h_{\text{mix}}(e_i))$ for the red, green, blue, and black coordinates $e_i \in \{(1,0,0),(0,1,0),(0,0,1),(0,0,0)\}$. We least-squares fit a single affine map $u \mapsto Wu + b$ that sends these four encoded mixtures to their RGB coordinates and apply it unchanged to every dense face sample. No other face points enter this alignment, so it removes the arbitrary chart convention without fitting the displayed surfaces. MSE compares aligned pixels with the reference faces; the null permutes the same pixels. Affine alignment can absorb axis scaling and shear, so the global-distance probe below tests Euclidean distance agreement separately.

For the decoder probe, we subtract the fixed carrier state $h_{\ell,p}$ from each decoded mixture and express the resulting displacement by least squares in the three directions obtained by similarly centering $D_\phi(E_\theta(a_{\text{red}}))$, $D_\phi(E_\theta(a_{\text{green}}))$, and $D_\phi(E_\theta(a_{\text{blue}}))$. These three coefficients are the plotted coordinates in the lower panel of Figure 5; no additional map is fitted to the dense samples. The same 64×64 face grids are compared with a permutation of the true points. Model-specific encoder and decoder panels use this common plotting protocol.

F.5 Replication of Table 3 for Other Models

We test whether the Qwen3-4B advantage in Table 3 remains stable across the other models. The per-model accuracies in Table 16 show that the structure-aligned interface exceeds both the direct hidden-state mixture and the linear color probe for every reported model. The spread across models is nevertheless substantial, from 29.7% on Qwen3-1.7B to 75.0% on Llama3-8B, so the aggregate result should not be read as uniform optimization behavior.

F.6 Robustness across intervention layers

The main result uses the three-quarter-depth layer; to test whether it depends on that single insertion point, we retrain the complete Qwen3-4B RGB interface independently at every layer and evaluate it on the same 64 representative colors. For reference, we also fit the linear color probe (Section F.1) at every layer, keeping its prompt, token position, and color split fixed.

The steerer succeeds at every insertion point (Figure 52). Its full-vocabulary accuracy ranges from 40.6% to 75.0%, whereas the direct hidden-state mixture stays at 1.6–4.7% and the linear color probe never exceeds 18.8%. Early insertion is the least stable: over layers 0–7, accuracy varies between 40.6% and 59.4%, whereas from layer 8 onward it stays between 51.6% and 75.0% (median 65.6%). The probe follows a different profile. It stays at or below 3.1% through layer 8, peaks at 18.8% in the middle layers, and falls back to 9.4–15.6% from layer 22 onward. At layers 0–6, where the probe is at most 3.1%, the steerer already reaches up to 59.4%, so its success does not require RGB geometry to be linearly readable from the model's own color-name states.

The encoder MSE stays below 0.05 at 34 of the 36 layers, rising only at layers 26 and 35 (up to 0.056), and the decoder MSE varies between 0.090 and 0.318 without a systematic depth trend. Even at those two layers, accuracy remains at 59.4% and 68.8%. The correlation analysis in Figure 53 makes this separation explicit: across layers, accuracy closely tracks the mean probability of the correct color token ($r = 0.83$) but is nearly uncorrelated with the encoder and decoder MSEs ($r = 0.05$ and -0.14). Thus the interface forms RGB-consistent coordinates at most depths, whereas the insertion layer mainly determines how readily the remaining layers convert them into the correct color token.

F.7 Replicating Figure 5 Across Models

We test whether the finding in Figure 5—that the encoder and decoder each learn geometry consistent with RGB task structure—remains stable across the other models. Figure 61 repeats its three-dimensional views for the seven additional models, and Figures 54 to 60 give their two-dimensional encoder faces. The result holds most cleanly for the four Llama models, whose encoders and decoders reproduce the joined faces with encoder MSE 0.005–0.014 and decoder MSE 0.023–0.033, against 0.2256 for permuted points. The three Qwen3 models reconstruct the cube less accurately but remain well below the same null: Qwen3-8B reaches 0.036 and 0.063, Qwen3-14B 0.087 and 0.118, and

Qwen3-1.7B 0.131 and 0.073. Their qualitative behavior separates them. Qwen3-8B and Qwen3-14B keep the three faces in both directions, with more curvature than the Llama models, and steer colors accurately (73.4% and 68.8%; Table 16). The decoded points of Qwen3-1.7B instead collapse onto nearly one direction despite its low decoder MSE, and it has the lowest RGB accuracy (29.7%). The cross-model result therefore supports a shared continuous organization through both directions of the interface, with reconstruction quality varying across models, and shows that a low distance-based error alone does not certify that the decoder preserves the cube's dimensions.

F.8 Two-dimensional view of the encoder faces

Figure 5 renders the three sampled RGB faces together in one three-dimensional space. Viewing each face on its own instead exposes channel-wise distortion, since varying two channels while holding the third fixed isolates how the encoder treats that pair of axes. Using the same 64×64 grids, output-anchor mixtures, and four-point affine alignment (Section F.4), the aligned encodings in Figure 62 reproduce the reference faces with mean MSE 0.0076, whereas permuting the same pixels raises it to 0.1120 ± 0.0002 ($p \ll 10^{-3}$). The faces are therefore recovered individually as well as in their joint arrangement. Section F.7 reports the corresponding views for the other models.

F.9 Replicating Figure 6 Across Models

We next test whether the global distance preservation in Figure 6 remains stable across the other models. For each model, we compare the encoded and RGB-space Euclidean distances of all 2,016 pairs of reference colors after fitting one shared scale. The results in Figures 63 to 69 yield Pearson correlations from 0.709 to 0.908. Thus the cross-model evidence extends beyond the appearance of selected cube faces to pairwise relations throughout the complete reference set.

F.10 Learning RGB Geometry under Coordinate-Region Holdout

The preceding probes show that the steerer organizes sampled mixtures by RGB geometry; a stronger test is whether this organization supports interpolation where no coordinate was observed during optimization. We therefore retrain the interface while rejecting every sampled training coordinate in the contiguous warm-color cuboid $[0.55, 1] \times [0.45, 1] \times [0, 0.55]$. Eleven reference colors lie in this region: yellow, orange, coral, salmon, gold, amber, tan, khaki, mustard, bronze, and copper. All 64 color tokens remain in the candidate set used to construct the target distribution during training; only coordinates in this region are excluded when sampling training mixtures. The experiment therefore withholds a continuous region of the input coordinate space rather than its associated output labels. In Figure 70, Qwen3-4B retains 45.5% accuracy in the omitted region, compared with zero for the direct hidden-state mixture, while reaching 66.0% on seen coordinates. This provides evidence that the learned interface partially interpolates its geometric correspondence into unobserved regions, although the seen–held-out gap rules out distribution invariance.

The same held-out-region evaluation for the remaining models appears in Figures 71 to 77. Held-out accuracy is 27.3% or 45.5%, remains below seen-region accuracy for every model, and exceeds the zero accuracy of the direct hidden-state mixture for all eight models. The result therefore supports partial interpolation across architectures without implying uniform generalization.

F.11 Replication of Figure 7 for Other Models

We test whether the coordinate-perturbation result in Figure 7 remains stable across the other models. Across every model in Figures 78 to 84, increasing the same inference-time Gaussian perturbation raises coordinate MSE and drives color accuracy toward zero. Although the initial accuracies and degradation rates differ, every model exhibits the same relationship between disrupted encoder geometry and degraded output behavior. The dependence of RGB steering on the learned coordinate organization is therefore not specific to the main Qwen3-4B run.

F.12 Replicating the RGB Controls in Table 4 Across Models

The preceding subsection reports the inference-time perturbations. Here, we repeat the training-time controls in Table 4 on the seven models beyond Qwen3-4B. We test how replacing task-

consistent RGB geometry with a nonlinear warp that alters metric relations or with one- and two-dimensional representations that collapse degrees of freedom affects steering, while holding the mixtures, architecture, and optimization fixed. Each control's coordinates are rescaled by the ratio between the mean pairwise distance among the true RGB reference coordinates and the mean pairwise distance among its own transformed coordinates. Without this matching, a warp that contracts distances would act like a lower soft-target temperature, which would confound a change of geometry with a change of τ. Model-specific results appear in Table 18. Across all eight models, task-consistent RGB training attains 27.3–69.6% continuous accuracy, compared with 6.4–32.4% for the nonlinear warp, 7.1–30.4% in two dimensions, and 4.7–7.9% in one dimension. Together with the inference-time results above, the consistent ordering shows that steering degrades both when learned coordinates are disrupted after training and when the interface is trained against geometry that changes RGB's distances or degrees of freedom. The connection between task-consistent geometry and output behavior therefore holds across the evaluated model families rather than only for the main Qwen3-4B run.

These accuracies score every control against the nearest color in true RGB space. For the warp, the nearest color changes for 56.5% of the evaluated coordinates, so even a steerer that realized its warped targets perfectly would score at most 43.5%. We therefore re-evaluate the saved checkpoints against each steerer's own training geometry, without retraining (Table 19). The warp control remains less accurate than the task-consistent steerer in every model, by 3.7 to 48.4 points. The warp therefore underperforms not only because its targets differ from true RGB, but also because the frozen model realizes them less readily.

G Supplementary Modular-Arithmetic Results

The main-text Modular experiment tests whether the circular interface developed above can support a practical computation rather than only represent task geometry. This appendix establishes parameter-matched comparisons and tests whether the resulting accuracy, continuous structure, and downstream operability persist across models and controls. Unless explicitly labeled as a 16-shot control, all results use the zero-shot prompt protocol and the pair-disjoint split in Section C.6; the application-specific operand prediction, exact group operation, and training procedure are specified in Section C.7.

G.1 Matched adaptation budgets

To determine whether the Modular gains come from task structure rather than a larger adaptation budget, we match the trainable scale of the learned solver, LoRA, and prefix tuning. Table 20 reports all counts normalized by residual width d; Ours includes both the deep encoder–decoder and branched-128 aligner. LoRA uses the smallest uniform integer rank over the selected attention and MLP projections whose count is at least $840d$, while layer-local direct K/V prefix tuning uses the smallest integer token count reaching the same threshold. Llama3-70B consequently requires 3,360 virtual tokens; reduced microbatching and gradient accumulation control its attention memory without changing the parameterization.

The complete Structure-Aligned solver uses $786.23d$–$837.91d$ parameters across models, while LoRA and prefix tuning use at least $840d$. The competing adaptation methods therefore have no smaller trainable budget, making the behavioral comparison below conservative with respect to parameter count.

G.2 Replicating Figure 8 Across Models

The main-text comparison establishes the practical advantage on Qwen3-4B; we first test whether it persists across model families under the same data and matched adaptation budgets. Every method uses the same 18,000 zero-shot training prompts and 500 pair-disjoint test prompts. The Structure-Aligned, LoRA, and prefix methods are trained end to end for their respective objectives, while logistic regression directly classifies the same extracted hidden states.

The per-model values in Table 21 show accuracies from 87.2% to 99.6%. The Structure-Aligned solver exceeds logistic regression, LoRA, and prefix tuning on every model, including both parameter-matched adaptation methods. Even the strongest LoRA result, 76.6% on Llama3-8B, remains below

the corresponding 98.2% Structure-Aligned result, and oracle decoding reaches 100% on all eight models, so the remaining system errors are operand and output-readability errors rather than failures to decode the result residue. The advantage therefore extends across model families and sizes without relying on a larger trainable budget, although no model reaches perfect accuracy under this one-epoch, operand-supervised protocol. Qwen3-1.7B is the one model whose steerer optimization is unstable across seeds; Section G.3 reports that spread and the criterion used to select its run.

G.3 Seed Stability on the Smallest Model

On Qwen3-1.7B the steerer reaches the same 100% oracle-anchor decoding as the larger models and a 93.8% complete-system accuracy, but its optimization is the least stable of the eight models. Across five training seeds the steerer–aligner system reaches 14.0%, 68.0%, 93.0%, 93.8%, and 14.4%, so two of the five runs collapse to chance ($1/7$).

Which runs collapse is visible before any evaluation. Checkpoint selection (Section B.7) scores steerer checkpoints by the full-vocabulary nearest-reference accuracy accumulated over 128 prompt-conditioned training examples, so it consults no validation or test example. The two collapsed runs select their best checkpoint at 0.23, whereas the other three reach 0.70, 0.91, and 0.92; for the standard run of every other model the same criterion lies between 0.95 and 0.98. A threshold of 0.5 therefore separates failed from successful fits across every run reported in this paper, using training examples alone. We treat a run below that threshold as a failed fit and, for Qwen3-1.7B, report the seed whose criterion is highest (0.92); no other model fell below the threshold, so none was retrained. The spread above is part of the result: on the smallest model this objective reaches the same accuracy as elsewhere, but reaches it less reliably.

G.4 Decoded States and Residue Anchors as Output Interfaces

The solver writes the decoded state $D_\phi(E_\theta(a_{\hat{r}}))$ of the selected residue $\hat{r}$ at the last prompt token of layer ℓ, and the remaining layers then produce the answer. To separate this output interface from operand decoding, we hold the Qwen3-4B aligner, its predicted residues, the 500 test prompts, the layer, and the token position fixed, and replace only the written state by the steerer's unembedding-derived residue anchor $a_{\hat{r}}$, where $\hat{r}$ is the residue predicted by the aligner. Both written states are also evaluated with the correct residue.

Table 22 shows that both written states make the frozen model output every correctly computed residue. With predicted residues, both reach 87.2%, exactly the aligner's result-residue accuracy, and they produce identical next-token predictions on all 500 prompts; with correct residues, both reach 100%. The remaining errors therefore arise entirely from operand decoding, whose residue accuracies are 96.4% for the first operand and 89.8% for the second. The steerer thus provides an interface through which the frozen model reliably outputs the computed answer. At these discrete residues, its decoded states and the anchors from which it is built are equally readable. This ablation compares only the seven exact residue endpoints: unlike direct anchor injection, the decoder also maps intermediate circular coordinates to hidden states, which we evaluate separately in Section G.7.

G.5 Robustness Across Layers and Arithmetic Settings

The cross-model result uses one selected intervention layer per model, so we next test whether the Qwen3-4B result depends on that insertion point. We start at layer 4 and sample one layer every four layers, obtaining layers $4, 8, 12, 16, 20, 24, 28$, and 32, which together span the early, middle, and late parts of the model. We retrain a complete layer-specific steerer and aligner at each sampled position. Figure 85 shows a pronounced depth dependence: layers 4 and 8 reach only 13.6% and 15.6%, whereas layers 12–32 range from 73.4% to 94.4%. Success is therefore not tied to the standard layer 27, but neither is it uniform across depth; the early hidden states do not yet support reliable operand readout under this protocol. Layer-local LoRA retrained at each sampled position behaves differently: it is the more accurate method at layers 4 and 8 (45.6% and 43.8%), but varies little with depth, so our system overtakes it from layer 12 onward and exceeds it, prefix tuning, and logistic regression at six of the eight sampled layers.

After testing robustness to where the interface is inserted, we next vary the range of absolute operands while retaining the same modular operation. We express every arithmetic setting in the common

task $p_k(a, b) =$ {a}+{b} mod {k}, the answer is with target $y = (a + b) \bmod k$. The standard task uses $a, b \in \{0, \ldots, 29\}$ and $k = 7$. For the operand-range sweep, we keep $k = 7$ but expand both operands to $\{0, \ldots, 39\}$ and $\{0, \ldots, 49\}$; the two aligner branches therefore predict among 40 or 50 absolute values before folding them into seven residue classes. We reconstruct the pair-disjoint split and retrain the complete steerer–aligner for each range. Figure 87 reports 91.0% and 94.2% accuracy, respectively, showing that strong performance persists as the absolute-operand vocabulary grows.

Having expanded the operand vocabulary at fixed $k = 7$, we next vary the cyclic group itself while retaining the standard operands $a, b \in \{0, \ldots, 29\}$. We retrain the complete system on $a + b \bmod 5$, $a + b \bmod 6$, $a + b \bmod 8$, and $a + b \bmod 9$. Each setting changes the number of residue anchors, their circular phases, and the size of the folded distributions and convolution. Accuracies remain between 93.2% and 95.4% across the four moduli (Figure 86). Five independent runs of the standard $a+b \bmod 7$ task range from 87.2% to 92.0%. Together with the operand-range results, these controls show stable strong performance across optimization runs, operand vocabularies, and cyclic-group sizes, while the layer sweep identifies early insertion as a clear failure mode.

G.6 Prompt-template transfer beyond the training form

The preceding robustness tests vary insertion depth, operand vocabulary, and modulus while retaining the standard query form. We next test robustness to a change in surface form without retraining. The standard pair-disjoint evaluation already tests new operand combinations but leaves their prompt template unchanged; here we hold the same held-out test-pair pool fixed and change only that template. For each form in Table 23 we cycle through the 184 held-out pairs to form 500 zero-shot prompts. As a diagnostic control, we also sample 500 distinct pairs from the 645-pair optimization pool and render them with the same templates, testing whether the surface-form shift remains difficult even for operand combinations used in training. Across the eight models and four new templates in Table 24, accuracy under the four new templates ranges from 11.6% to 52.2% on held-out test pairs, averaging 30.4%, and averages 34.9% on optimization pairs. The original template instead averages 95.5% and 99.7% on the same two pair sources. Because the trained steerer, aligner, and arithmetic composition are unchanged, the much larger template effect than pair-split effect localizes the main limitation to predicting the operands from the altered prompt state or making the computed residue the model's top prediction in the altered carrier context.

G.7 Continuous modular readout between residue anchors

The encoding analysis in Figure 98 establishes circular organization, but not whether intermediate coordinates produce the intended output distributions after the remaining layers. We therefore sample 50 phases on each neighboring-residue arc and evaluate each phase under 16 zero-shot operand-pair prompts from the test split. The learned condition inserts $h_{\text{steer}}(z) = D_\phi(E_\theta(h_{\text{mix}}(z)))$; direct interpolation inserts $h_{\text{mix}}(z)$; and nearest-anchor snapping inserts $a_{j^*(z)}$, where $j^*(z) = arg\,min_j d_T(z, c_j)$. For each method, we average the resulting digit distributions across prompts and measure total-variation error from the geometry-derived target $q(z)$.

On Qwen3-4B, Figure 88 shows clearly lower mean error for the Modular steerer (0.139) than for direct interpolation (0.220), with an advantage across most phases, and a small margin over nearest-anchor snapping (0.145). Using the same 350 phases and prompt protocol, the additional-model results in Figures 89 to 95 show that the steerer has lower error than direct interpolation for every model. Nearest-anchor snapping remains better for Llama3.1-8B and Llama3-70B, but it replaces every intermediate phase with one of the seven discrete anchors: its state is constant within each nearest-anchor region and changes abruptly at the region boundary. It therefore does not provide a continuous readout of intermediate task coordinates. The steerer combines such a continuous readout with lower error throughout the Qwen family, while its advantage over both baselines is less uniform across the Llama models; discrete-answer accuracy alone would conceal this variation.

G.8 Evaluation Protocol for the Additional $\mathbb{Z}_7$ Action

We evaluate the additional action in Table 5 on the same 500 held-out zero-shot prompts as the standard Modular result. For each $p(a, b)$, the standard procedure in Section C.7 selects $\widehat{r}$ from the aligner's result distribution and inserts $D_\phi(z_{\widehat{r}})$, with target $(a + b) \bmod 7$.

For the additional-action condition, we cycle through independently shuffled copies of all 49 ordered pairs $(c, d) \in \mathbb{Z}_7^2$ and truncate the sequence to the 500 test examples, so each pair occurs 10 or 11 times. With $\Delta = (c + d) \bmod 7$, we replace the same hidden state with $D_\phi(z_{(\widehat{r}+\Delta) \bmod 7})$ and run the remaining layers once, now targeting $(a + b + c + d) \bmod 7$. Thus the intervention applies one exact discrete $\mathbb{Z}_7$ action in the learned anchor frame; it does not decode and re-encode an intermediate token. The oracle control uses the true residue $r^* = (a + b) \bmod 7$ in place of $\widehat{r}$. Original accuracy and accuracy after the action are both 87.2%, while oracle-action accuracy is 100%. The action therefore adds no error in this setting. This protocol tests one additional action followed by one decode, not repeated decode–encode reuse or unrestricted latent computation.

G.9 Qwen3-4B 16-shot control

The standard results use zero-shot queries, so we test on Qwen3-4B whether the same design remains effective when demonstrations substantially change the carrier context. Each prompt draws 16 distinct ordered operand pairs without replacement from the optimization-pair pool and formats each as a solved line of the form `a+b mod 7, the answer is r`, where $r = (a + b) \bmod 7$. These demonstrations are followed by one unanswered query in the same format. The query pair is drawn from the optimization, validation, or test pool according to the split being constructed, whereas demonstrations always come from the optimization pool; because (a, b) and (b, a) are assigned to the same split, a held-out query (a, b) can contain neither (a, b) nor (b, a) among its demonstrations. For an optimization-split query, we exclude the identical ordered pair (a, b) from its demonstration candidates; the reversed pair (b, a) is not additionally excluded. The 16 demonstrations are sampled independently for each prompt. We train a separate steerer and aligner on these 16-shot contexts.

Figure 96 compares the separately trained pipelines and baselines under zero- and 16-shot prompting. In the 16-shot condition, the Structure-Aligned system reaches 82.2%, compared with 51.0% for LoRA, 10.8% for prefix tuning, 6.8% for logistic regression, and 27.6% for the unmodified LLM. It therefore retains most of its zero-shot accuracy and remains substantially more accurate than both the parameter-matched adaptation methods and the simpler baselines.

Prompt-level accuracy does not show whether the continuous circular behavior in Figure 88 survives the longer demonstration context. We therefore repeat that figure's protocol using the separately trained 16-shot steerer: we sample 50 phases on each neighboring-residue arc, decode them under 16 held-out 16-shot carrier prompts, average the induced digit distributions, and compute phasewise TV error from the same geometry-derived targets. Direct interpolation and nearest-anchor snapping use the same phases, prompts, and remaining layers. In Figure 97, the 16-shot steerer has mean TV error 0.096, compared with 0.354 for direct interpolation and 0.260 for nearest-anchor snapping. The steerer therefore retains its continuous circular output behavior when the carrier includes demonstrations, rather than only preserving accuracy at the seven discrete residues.

To determine whether the remaining 16-shot errors are amplified by further coordinate computation, we repeat the additional-action diagnostic above. Table 25 shows identical 82.2% accuracy before and after the action, while the oracle remains perfect. The action therefore adds no new behavioral errors: the observed failures are already present before the extra action.

G.10 Modular encoding geometry across models

We test whether a Modular steerer trained across arithmetic prompts recovers the circular structure it shares with Number. For each model, 50 phases are sampled on every neighboring arc, mixtures are constructed from all seven Modular anchors, and the resulting 350 encodings are aligned to the reference circle by one orthogonal transform fitted on the exact residue phases. The matched null permutes the same circle coordinates across phases 100 times and receives the identical alignment.

Together, Figure 98 and Figures 99 to 105 show that the Modular steerer recovers the circular structure in most models. Seven of eight learned trajectories closely trace the closed reference circle

(MSE $= 0.005$–0.043), far below the distribution-matched null (0.995 ± 0.037). Qwen3-8B largely follows the circle as well but deviates locally (Figure 104): it exchanges the positions of residues 1 and 6, while the remaining residues keep their cyclic order. Because the encoder output is normalized, every encoding still lies on the circle; the exchanged residues only make a few interpolated arcs jump between distant points, which appear as chords in the figure. This local deviation raises the aligned MSE to 0.470, still less than half of the null, and leaves arithmetic accuracy intact: Qwen3-8B reaches 95.8%, close to the 94.8% of Qwen3-14B, whose realization stays ordered. Thus the Modular interface recovers the circular structure it shares with Number closely in seven models and, apart from a few arcs, in the remaining one.

H Matched Comparison with Manifold Steering

This section details the matched setting, method construction, and evaluation behind the comparison with Manifold Steering reported in Table 1. Following its weekday setting (Wurgaft et al., 2026), we use Llama-3.1-8B at layer 28 and the question format

```
Q: What day is {k} days after {w}?
A:
```

where k is a number from one to seven and w a weekday, so the answer is $(w + k) \bmod 7$. Taking every combination yields 49 corpus examples. Let h_i be the last-token layer-28 state, y_i its answer weekday, and μ the corpus mean. If the rows of $U \in \mathbb{R}^{r\times d}$ are the top r principal directions, the fitted PCA coordinates are $P(h) = U(h-\mu)$. We follow their PCA-64 setting; the 49 corpus examples cap the fitted subspace at $r = 48$. Manifold Steering forms each weekday centroid $a_j = \frac{1}{7}\sum_{i:y_i=j} h_i$ and determines their cyclic order from the angles of the variance-normalized first two PCA coordinates, then fits a periodic cubic spline γ through the full PCA centroids at those angular positions. For a carrier state h_p and a position u on the source–target spline arc, the intervention replaces its PCA coordinates while preserving its component outside the fitted subspace:

$$\widetilde{h}_p(u) = h_p + U^\top[\gamma(u) - P(h_p)]\,, \tag{43}$$

where $\gamma(u) \in \mathbb{R}^r$. Writing $h_p = \mu + U^\top P(h_p) + r_p^\perp$ shows that this replacement changes only the PCA coordinates: $\widetilde{h}_p(u) = \mu + U^\top\gamma(u) + r_p^\perp$ retains the carrier residual $r_p^\perp$ outside the fitted subspace. The 16 evaluation carriers are sampled with replacement from these 49 corpus prompts.

To adapt our steerer to this setting, we replace its standard unembedding-based Weekday anchors with the same seven empirical centroids a_j. We directly train the encoder to place the centroids and their interpolations at the prescribed weekday phases $c(j)$ and train the decoder to return model-readable states. A source–target intervention then decodes the shortest circular arc between $c(s)$ and $c(t)$. Thus both methods use the same corpus-derived anchors and evaluation carriers; they differ in whether paths are fitted in hidden-state PCA space or generated by a shared cyclic coordinate interface.

Evaluating every one of the $\binom{7}{2} = 21$ unordered source–target pairs prevents performance on a selected path from determining the comparison, while 50 equally spaced waypoints test whether each path preserves intermediate weekday order rather than merely reaching its endpoint. Under all 16 carriers, let $\widehat{y}_{s,t,p,n}$ denote the full-vocabulary top prediction at waypoint n, and let $y^*_{s,t,n}$ be the weekday nearest the corresponding position on the intended shortest arc. Final-target and ordered-path accuracy are respectively

$$A_{\text{end}} = \frac{1}{21\cdot 16}\sum_{s<t,p}\mathbf{1}[\widehat{y}_{s,t,p,50} = t], \qquad A_{\text{path}} = \frac{1}{21\cdot 16\cdot 50}\sum_{s<t,p,n}\mathbf{1}[\widehat{y}_{s,t,p,n} = y^*_{s,t,n}]. \tag{44}$$

To evaluate composition, we define one reusable day-advance map for each method rather than concatenate separately fitted pairwise paths. Ours uses $T_{\text{ours}}(h) = D_\phi(R_{2\pi/7}E_\theta(h))$. For Manifold Steering, let $u(h)$ be the coordinate of the spline point nearest $P(h)$ and let Δu be the signed periodic displacement from the Monday centroid to the Tuesday centroid. Its map is $T_{\text{MS}}(h) = h + U^\top[\gamma(u(h) + \Delta u) - P(h)]$. We start from the Monday state and apply the corresponding map seven times. If $\widehat{y}_{p,k}$ is the prediction after k applications, repeated-step accuracy averages $\mathbf{1}[\widehat{y}_{p,k} = k \bmod 7]$ over all 16 carriers and $k = 0,\ldots,7$, whereas cycle-return accuracy averages only $\mathbf{1}[\widehat{y}_{p,7} = \text{Monday}]$. These metrics distinguish a correct endpoint, semantic order along individual

paths, and reuse of one operation throughout a complete cycle. The corresponding results are reported in Table 1.

I Comparison with a Circular-Subspace Intervention

Section H compares our steerer with a method that builds a path between behavior centroids in hidden-state space. This section compares it with a second nonlinear baseline of a different kind: the circular-subspace intervention of Engels et al. (2025), which writes into a subspace that already carries a circular weekday feature. We keep the model, corpus, and carriers of Section H.

I.1 Construction

Manifold Steering and our steerer replace the final prompt-token state of the prompts in Section H, which produces the answer. This intervention instead replaces the state of their weekday token w: writing the circle point of another weekday w' there resets the question's source day, and the frozen model still performs the arithmetic. Every state below is taken at that token.

Let $X \in \mathbb{R}^{49\times d}$ collect the corpus states at the chosen layer, μ their mean, and $W \in \mathbb{R}^{m\times d}$ the top $m = 5$ principal directions of $X - \mu$. A least-squares probe $P \in \mathbb{R}^{2\times m}$ maps the principal coordinates to the reference circle, $PW(x-\mu) \approx c(w)$ with $c(j) = (\cos\frac{2\pi j}{7}, \sin\frac{2\pi j}{7})$. Writing a circle point z replaces the fitted subspace and mean-ablates the rest of the state,

$$x^* = \mu + W^\top P^+ z, \qquad \widehat{z}(h) = PW(h-\mu), \tag{45}$$

with P^+ the pseudoinverse; $\widehat{z}$ reads the current circle point back. All methods in this comparison are fitted on the same 49 states, and the 16 carriers are drawn from those same prompts, so the comparison controls the fitting data rather than testing generalization to unseen prompts. For the comparison we refit our steerer at the same token, training it as in Section H.

Because $PP^+ = I_2$, Equation (45) is read back exactly: $\widehat{z}(x^*) = z$. Any map of the form "read the current circle point, rotate it, write it back" therefore composes and closes algebraically, and seven applications return the starting state to numerical precision (relative L_2 below 10^{-6}). The composition metrics of Table 1 would then record the accuracy of the starting state rather than a property of the intervention. Section I.2 therefore evaluates both methods with metrics suited to an intervention that sets an operand.

I.2 Evaluation at the operand-token site

Let $\widehat{y}(p, w')$ be the model's full-vocabulary top prediction after writing $c(w')$ at the weekday token of carrier p, whose own prompt uses the number k_p and the weekday w_p. The first metric is answer accuracy: a correct intervention makes the model answer $(w' + k_p) \bmod 7$, and we report the fraction of the 16×7 carrier–weekday pairs for which it does. We also evaluate the unmodified model on the matched natural prompt containing weekday w' and number k_p. We call the pairs it answers correctly *base-solvable* and report the intervention's answer accuracy on this subset separately.

We further report the metric used by Engels et al. (2025), the log-probability the intervention moves from the original answer $\gamma_p = (w_p + k_p) \bmod 7$ to the written one. Only the six weekdays that differ from the carrier's own change the question, so with p_γ the probability aggregated over the spellings of weekday γ,

$$\Delta \log p = \frac{1}{16\cdot 6} \sum_{p} \sum_{w' \neq w_p} \left[\log p_{(w'+k_p) \bmod 7} - \log p_{\gamma_p}\right], \tag{46}$$

which is positive when the written weekday becomes the more probable answer. Finally, a fractional sweep writes $c(w' + u)$ for $u \in \{0, \frac{1}{20}, \ldots, 7 - \frac{1}{20}\}$ and counts a sample correct when the model answers $(\lfloor w' + u \rceil + k_p) \bmod 7$, where $\lfloor\cdot\rceil$ rounds to the nearest weekday. Its accuracy measures whether intermediate circle points are read as intermediate weekdays rather than snapping to one of the seven anchors.

Table 26 reports these quantities for both methods. Through layer 16 the intervention is effective and our interface leads on every metric: its answer accuracy is 95.5%–96.4%, against 73.2%–78.6% for

the subspace intervention, and is correct on every base-solvable pair, which the subspace intervention reaches for at most 80.8%. It also exceeds the model's own 92.9% on these questions, so writing a clean weekday state additionally repairs some of its errors. The fractional sweep separates the two further: our decoded states follow the intended phase for 85.0%–87.9% of samples, against 66.4%–67.7% for the subspace intervention, so intermediate circle points are more often read as the intermediate weekday rather than snapping to an anchor.

At layers 20 and 24 both methods collapse to roughly chance: answer accuracy falls to at most 21.4%, phase-following accuracy to at most .194 against a chance level of $1/7$, and $\Delta \log p$ turns negative, so the original answer remains preferred. The probe still fits the corpus at these depths ($R^2 \geq 0.98$), so the failure is not an absent circle but an intervention site that no longer drives the answer; Engels et al. (2025) report the same decline once the source weekday has been copied to the final token. These two layers therefore bound where an operand-site intervention of either kind is informative, rather than separating the methods.

J Steerer Capacity Across Task Geometries

The capacity result in Section 3.2 establishes that a sufficiently expressive interface can realize the required mappings, but does not determine how much nonlinear depth the empirical tasks require. We test this by replacing the standard encoder and decoder with matched shallow variants that remove their middle $256 \to 256$ layers, while holding the model, intervention layer, data, objective, training protocol, and random seed fixed (architectures in Section B.4). In Figure 106, Qwen3-4B RGB accuracy falls from 75.0% to 21.9%, whereas Number changes only from 98.0% to 94.5% and Modular from 87.2% to 87.0%. The same reduction in interface depth therefore has a much larger behavioral effect for RGB's higher-dimensional, more complex task space than for the one-dimensional circular structures shared by Number and Modular. This contrast is consistent with the three-dimensional RGB correspondence requiring greater approximation capacity, while showing that the strong Number and Modular results do not merely depend on using the deepest tested architecture.

Table 7: Models and standard intervention layers. Qwen3-4B is the primary analysis model; the remaining models provide cross-model replications.

Model	Transformer layers L	Standard layer ℓ
Llama3-8B	32	24
Llama3.1-8B	32	24
Llama3.2-3B	28	21
Llama3-70B	80	60
Qwen3-1.7B	28	21
Qwen3-4B	36	27
Qwen3-8B	36	27
Qwen3-14B	40	30

Table 8: Output-anchor accuracy when either the target or carrier semantics are randomized. Random Target uses 512 distinct non-special target tokens with the fixed `Today is` carrier; Random Carrier averages the seven weekday targets over 50 random-token carriers.

Model	Random Target	Random Carrier
Llama3-8B	.992	1.000
Llama3.1-8B	.992	1.000
Llama3.2-3B	.992	1.000
Llama3-70B	.990	1.000
Qwen3-1.7B	.000	.146
Qwen3-4B	.918	.994
Qwen3-8B	.895	.974
Qwen3-14B	.674	.897

Table 9: Default steerer optimization settings. "Full-vocabulary soft CE" denotes the geometry-matching objective in Equation (8).

Task	Steps	Batch	Learning rate	Weight decay	Active losses
Weekday	2000	full set	5×10^{-3}	0	coordinate + reconstruction
Number	2000	64	5×10^{-4}	10^{-2}	full-vocabulary soft CE
RGB	4000	64	1×10^{-4}	10^{-2}	full-vocabulary soft CE
Modular	2000	16	5×10^{-4}	10^{-2}	full-vocabulary soft CE

Table 10: Number task elements and circular coordinates.

Task element s	Phase	$c(s)$
0	0	$(1, 0)$
1	$2\pi/7$	$(\cos(2\pi/7), \sin(2\pi/7))$
2	$4\pi/7$	$(\cos(4\pi/7), \sin(4\pi/7))$
3	$6\pi/7$	$(\cos(6\pi/7), \sin(6\pi/7))$
4	$8\pi/7$	$(\cos(8\pi/7), \sin(8\pi/7))$
5	$10\pi/7$	$(\cos(10\pi/7), \sin(10\pi/7))$
6	$12\pi/7$	$(\cos(12\pi/7), \sin(12\pi/7))$

Table 11: RGB task elements and coordinates. For a color name represented by multiple tokens, its first token is used as the output label.

Element	(R, G, B)	Element	(R, G, B)	Element	(R, G, B)	Element	(R, G, B)
black	$(0, 0, 0)$	azure	$(0, .50, 1)$	crimson	$(.86, .08, .24)$	jade	$(0, .66, .42)$
white	$(1, 1, 1)$	brown	$(.59, .29, 0)$	scarlet	$(1, .14, 0)$	chartreuse	$(.50, 1, 0)$
red	$(1, 0, 0)$	coral	$(1, .50, .31)$	burgundy	$(.50, 0, .13)$	spring	$(0, 1, .50)$
green	$(0, 1, 0)$	turquoise	$(.25, .88, .82)$	salmon	$(.98, .50, .45)$	sky	$(.53, .81, .92)$
blue	$(0, 0, 1)$	aqua	$(0, .75, .75)$	peach	$(1, .85, .73)$	cerulean	$(0, .48, .65)$
yellow	$(1, 1, 0)$	fuchsia	$(.80, 0, .60)$	gold	$(1, .84, 0)$	sapphire	$(.06, .32, .73)$
cyan	$(0, 1, 1)$	lime	$(.50, 1, .20)$	amber	$(1, .75, 0)$	cobalt	$(0, .28, .67)$
magenta	$(1, 0, 1)$	violet	$(.56, 0, 1)$	apricot	$(.98, .81, .69)$	royal	$(.25, .41, .88)$
grey	$(.50, .50, .50)$	indigo	$(.29, 0, .51)$	tan	$(.82, .71, .55)$	aquamarine	$(.50, 1, .83)$
silver	$(.75, .75, .75)$	lavender	$(.90, .90, .98)$	beige	$(.96, .96, .86)$	mustard	$(1, .86, .35)$
purple	$(.50, 0, .50)$	plum	$(.87, .63, .87)$	khaki	$(.94, .90, .55)$	charcoal	$(.21, .27, .31)$
teal	$(0, .50, .50)$	orchid	$(.85, .44, .84)$	ivory	$(1, 1, .94)$	slate	$(.44, .50, .56)$
olive	$(.50, .50, 0)$	mauve	$(.88, .69, 1)$	cream	$(1, .99, .82)$	chocolate	$(.48, .25, 0)$
maroon	$(.50, 0, 0)$	periwinkle	$(.80, .80, 1)$	mint	$(.60, 1, .60)$	bronze	$(.80, .50, .20)$
navy	$(0, 0, .50)$	pink	$(1, .40, .70)$	emerald	$(0, .79, .34)$	copper	$(.72, .45, .20)$
orange	$(1, .50, 0)$	rose	$(1, 0, .50)$	forest	$(.13, .55, .13)$	sepia	$(.44, .26, .08)$

Table 12: Canonical Modular evaluation protocol. The validation pairs and examples are held out from training but are not used by the reported protocol (Section C.6).

Setting	Value
Modulus	7
Operand range	0–29 (inclusive)
Prompt demonstrations	0
Optimization/validation/test examples	18,000 / 500 / 500
Outer test-pair fraction	0.2
Inner validation-pair fraction	0.1
Pair constraint	result-stratified, swap-orbit disjoint
Evaluation	full-vocabulary next-token top-1

Table 13: On Qwen3-4B, end-to-end output supervision trains an accurate Weekday steerer without coordinate or hidden-state targets. Dense accuracy evaluates 512 randomly sampled phases; exact accuracy evaluates the seven weekday phases.

Model	End-to-End	Hidden-State Mixture	Linear Probe	Exact Label
Qwen3-4B	.965	.414	.289	1.000

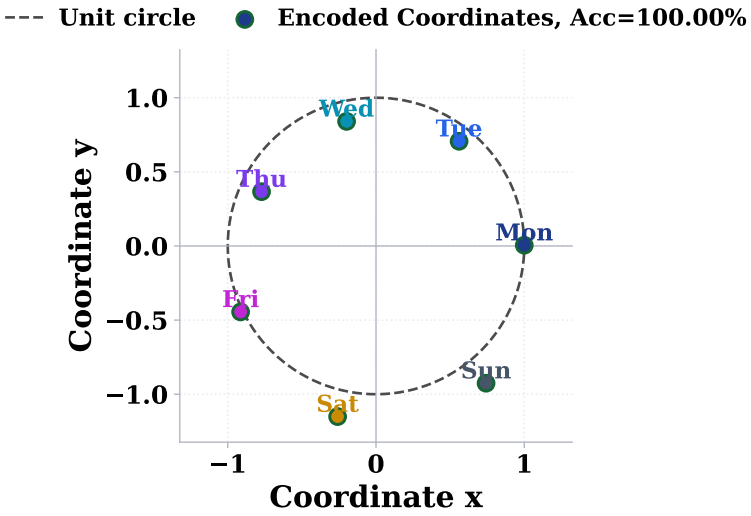


Figure 9: Direct supervision organizes the Qwen3-4B Weekday states into the prescribed circle, and green outlines show that all seven decoded outputs are correct.

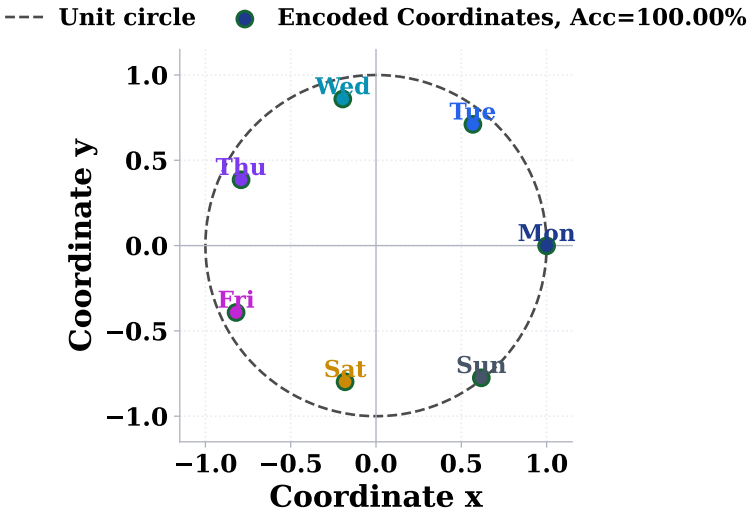


Figure 10: Direct supervision organizes the Llama3-8B Weekday states into the prescribed circle, and green outlines show that all seven decoded outputs are correct.

Table 14: Number accuracy (%) over densely sampled phases for every model. The direct hidden-state mixture inserts the anchor mixture without the steerer; linear regression fits one affine map from mixtures to circular coordinates. Qwen3-4B reports the mean and standard deviation over five training seeds, as in Table 2; every other model reports one run.

Model	Steerer	Hidden-state mixture	Linear regression
Llama3-8B	**94.7**	38.5	19.9
Llama3.1-8B	**88.1**	38.5	12.9
Llama3.2-3B	**95.9**	42.8	13.1
Llama3-70B	**95.3**	37.1	23.6
Qwen3-1.7B	**87.9**	18.9	13.1
Qwen3-4B	**97.9±1.2**	41.8±1.6	17.4±3.5
Qwen3-8B	**94.9**	40.4	14.8
Qwen3-14B	**96.3**	39.6	29.9

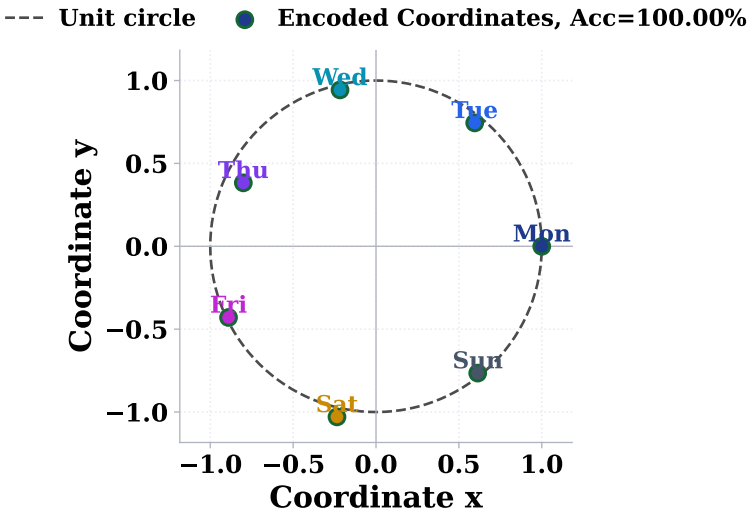


Figure 11: Direct supervision organizes the Llama3.1-8B Weekday states into the prescribed circle, and green outlines show that all seven decoded outputs are correct.

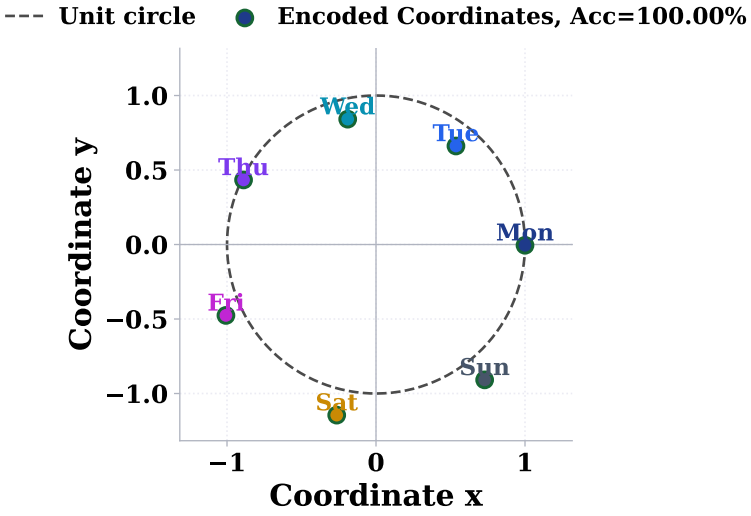


Figure 12: Direct supervision organizes the Llama3.2-3B Weekday states into the prescribed circle, and green outlines show that all seven decoded outputs are correct.

Table 15: Number steerers trained on the task-consistent circle, a permuted circle, or a line. Dense accuracy (%) scores the nearest digit over densely sampled phases; closure error is the seven-step relative L_2 error of the repeated one-step operator.

	Dense accuracy			Seven-step closure error		
Model	Circle	Permuted	Line	Circle	Permuted	Line
Llama3-8B	**94.9**	72.7	82.4	0.678	0.841	1.295
Llama3.1-8B	**88.9**	71.7	84.3	0.598	0.827	1.584
Llama3.2-3B	**96.6**	64.7	84.9	0.401	1.066	1.318
Llama3-70B	**95.0**	74.4	84.6	1.077	0.989	1.755
Qwen3-1.7B	**87.4**	74.0	80.1	0.557	1.082	1.509
Qwen3-4B	**96.7**	73.6	83.1	0.707	0.884	1.677
Qwen3-8B	**95.4**	72.9	84.7	0.982	0.972	1.137
Qwen3-14B	**96.1**	72.0	83.1	0.458	0.951	1.473

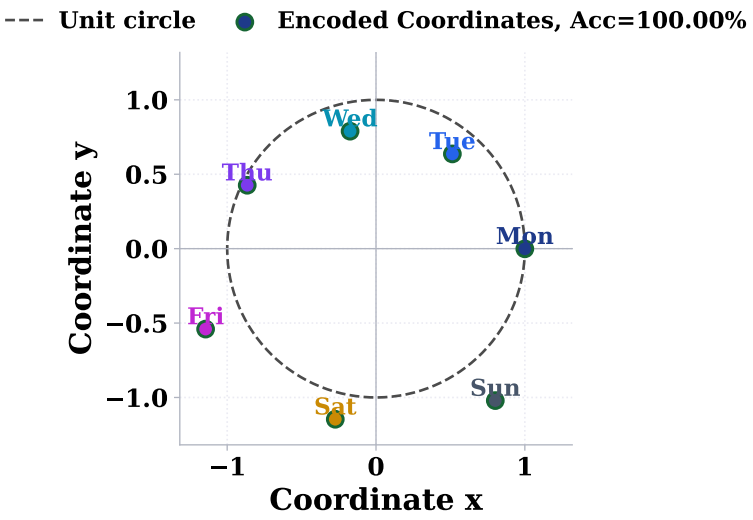


Figure 13: Direct supervision organizes the Llama3-70B Weekday states into the prescribed circle, and green outlines show that all seven decoded outputs are correct.

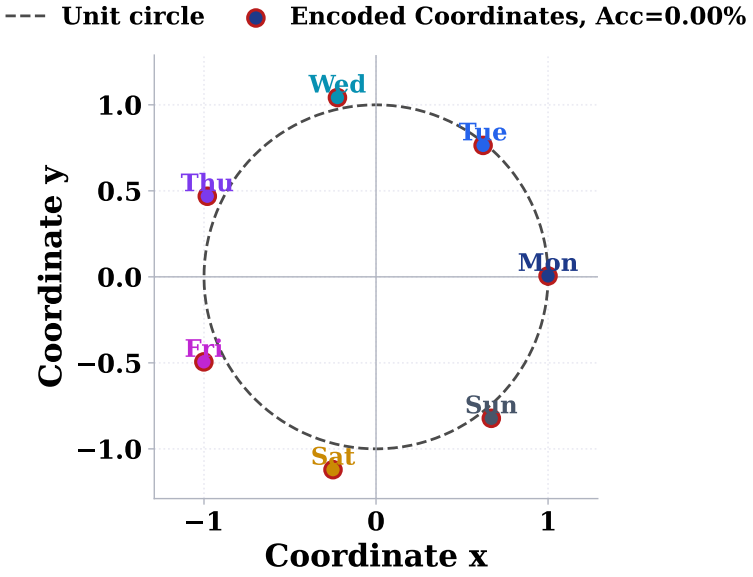


Figure 14: Qwen3-1.7B learns the prescribed circular coordinates, but red outlines show that none of the seven decoded states produces its intended Weekday output at the default anchor scale.

Table 16: RGB accuracy (%) on the 64 representative colors for every model. The direct hidden-state mixture inserts the raw anchor mixture; the linear color probe fits an affine map from color-name states to RGB coordinates. Qwen3-4B reports the mean and standard deviation over five training seeds, as in Table 3; every other model reports one run.

Model	Steerer	Hidden-state mixture	Linear color probe
Llama3-8B	**75.0**	4.7	18.8
Llama3.1-8B	**73.4**	4.7	18.8
Llama3.2-3B	**60.9**	4.7	18.8
Llama3-70B	**67.2**	4.7	9.4
Qwen3-1.7B	**29.7**	0.0	9.4
Qwen3-4B	**70.3±9.8**	4.7±0.0	9.4±0.0
Qwen3-8B	**73.4**	4.7	9.4
Qwen3-14B	**68.8**	0.0	12.5

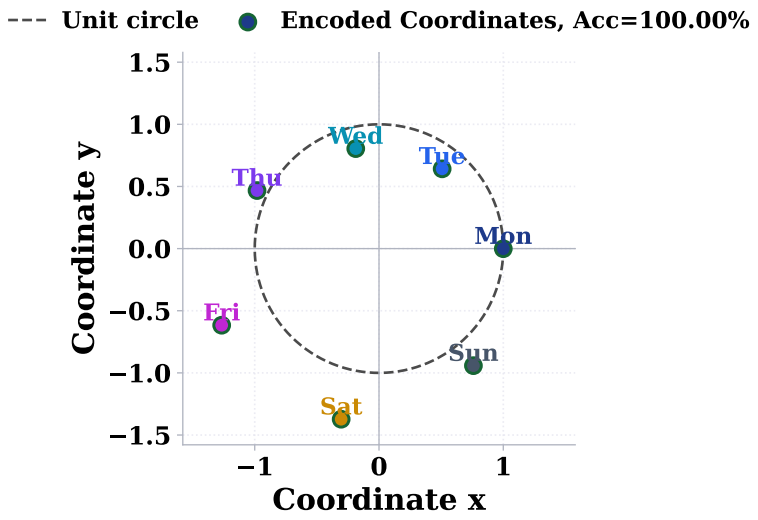


Figure 15: Direct supervision organizes the Qwen3-8B Weekday states into the prescribed circle, and green outlines show that all seven decoded outputs are correct.

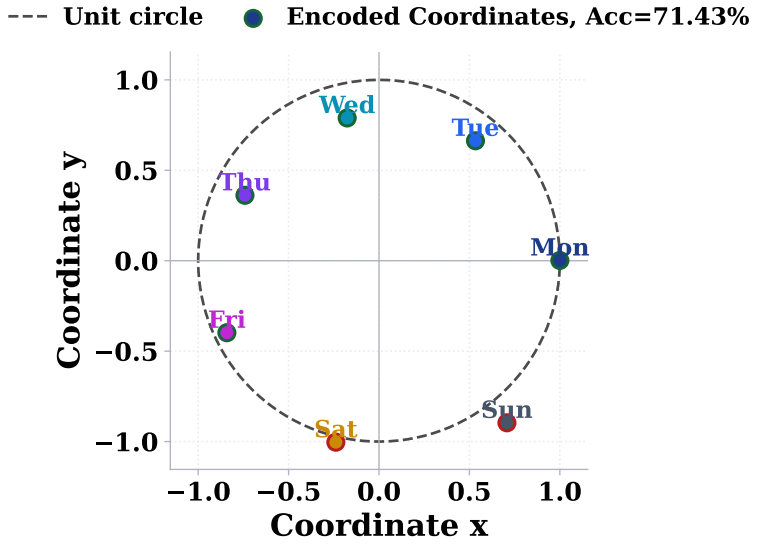


Figure 16: Qwen3-14B learns the prescribed circular coordinates, but green and red outlines show that only five of the seven decoded states produce their intended Weekday outputs at the default anchor scale.

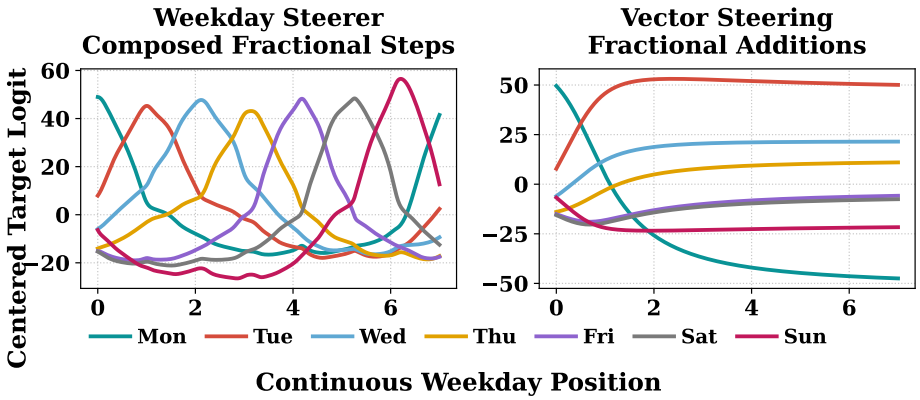


Figure 17: On Llama3-8B, the circular interface follows the full weekday cycle more faithfully than repeated fixed-vector addition.

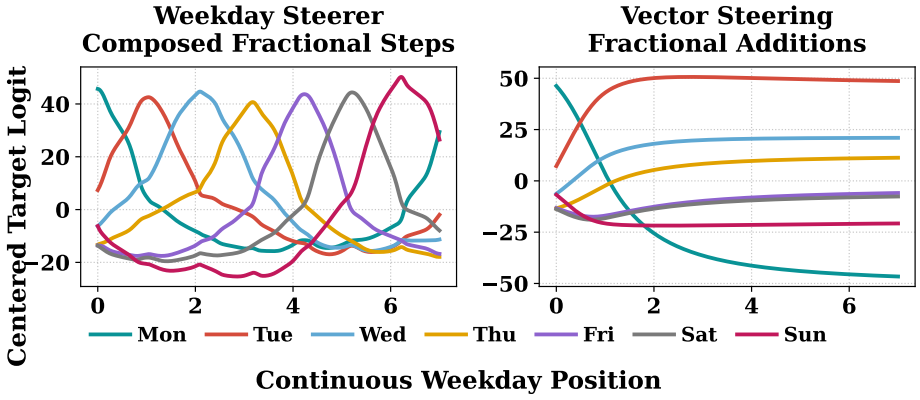


Figure 18: On Llama3.1-8B, the circular interface follows the full weekday cycle more faithfully than repeated fixed-vector addition.

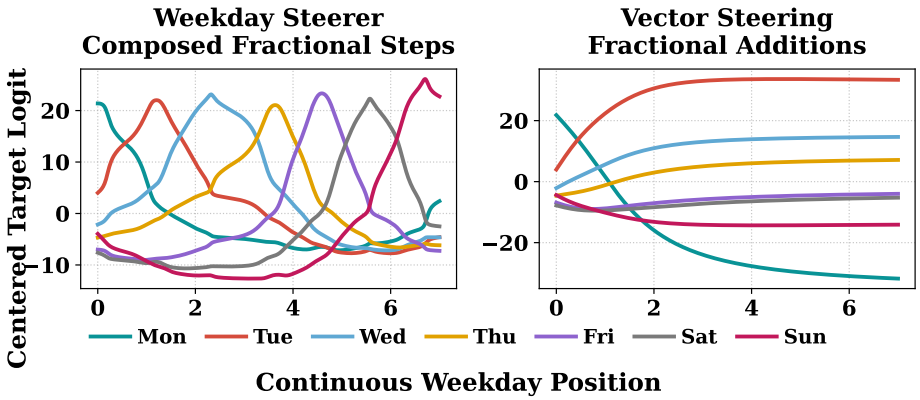


Figure 19: On Llama3.2-3B, the circular interface follows the full weekday cycle more faithfully than repeated fixed-vector addition.

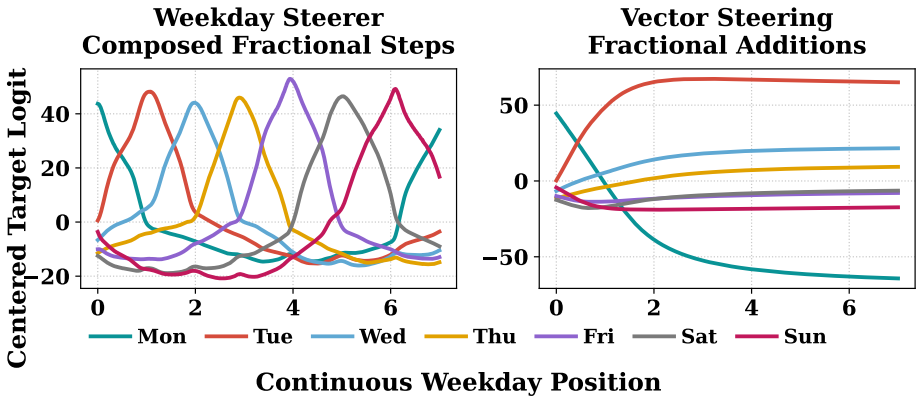


Figure 20: On Llama3-70B, the circular interface follows the full weekday cycle more faithfully than repeated fixed-vector addition.

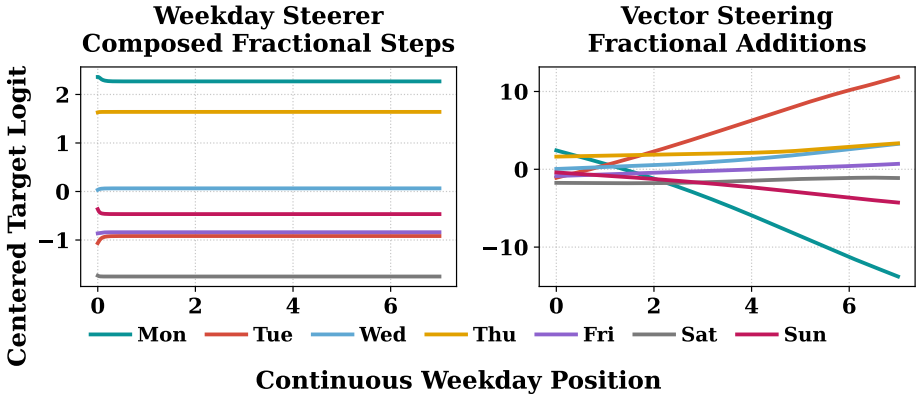


Figure 21: On Qwen3-1.7B, the circular interface follows the full weekday cycle more faithfully than repeated fixed-vector addition.

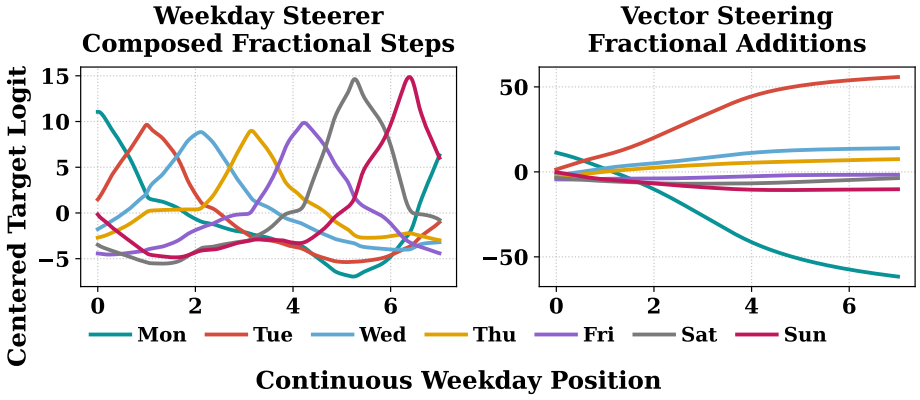


Figure 22: On Qwen3-8B, the circular interface follows the full weekday cycle more faithfully than repeated fixed-vector addition.

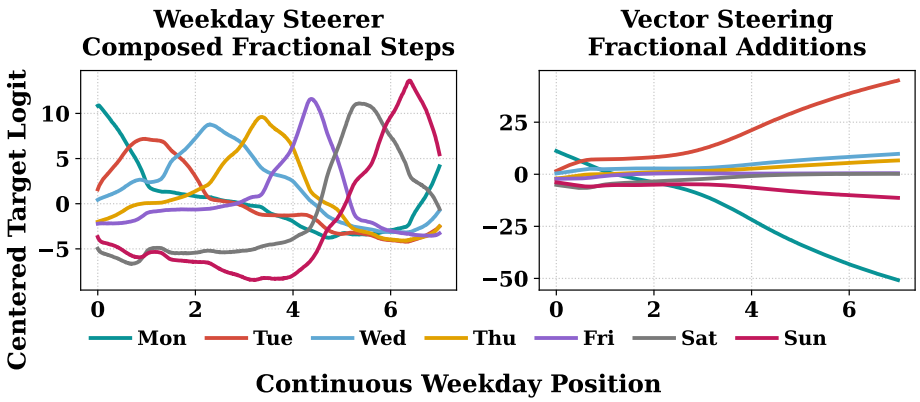


Figure 23: On Qwen3-14B, the circular interface follows the full weekday cycle more faithfully than repeated fixed-vector addition.

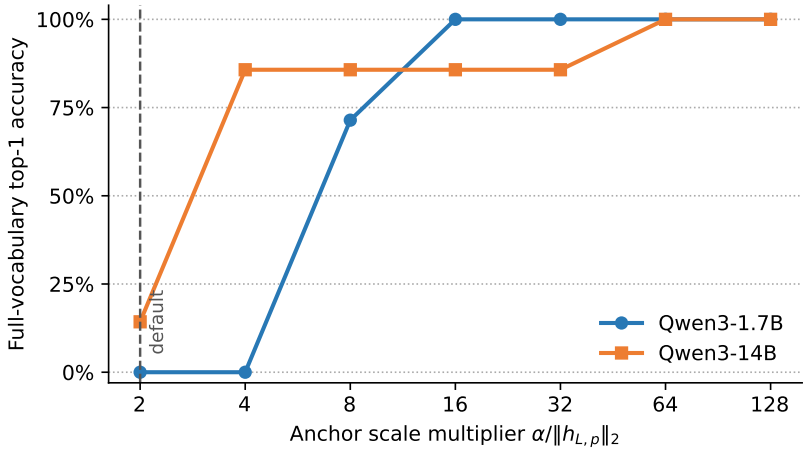


Figure 24: Increasing the unembedding intervention scale restores full-vocabulary readability of all seven weekday anchors: Qwen3-1.7B reaches 100% at $16\|h_{L,p}\|_2$, and Qwen3-14B reaches 100% at $64\|h_{L,p}\|_2$.

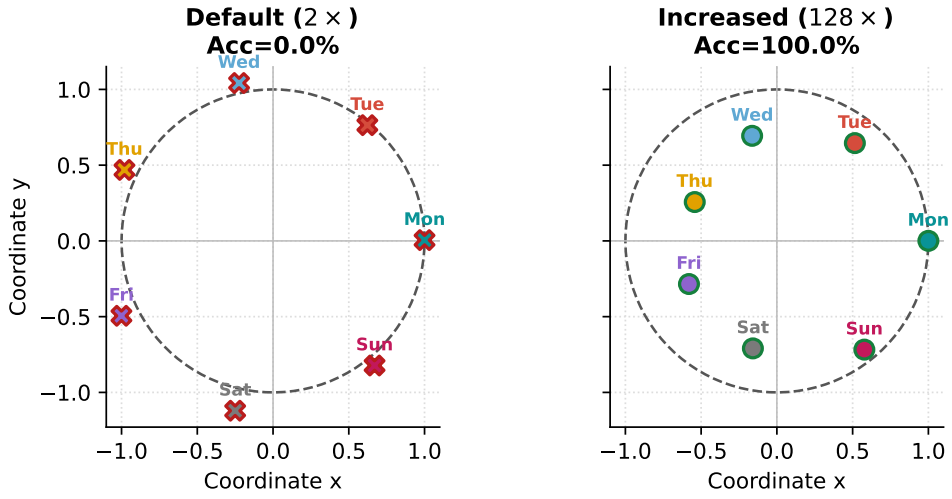


Figure 25: Increasing the Qwen3-1.7B weekday-anchor scale restores all seven discrete steering outputs while preserving the learned circular organization.

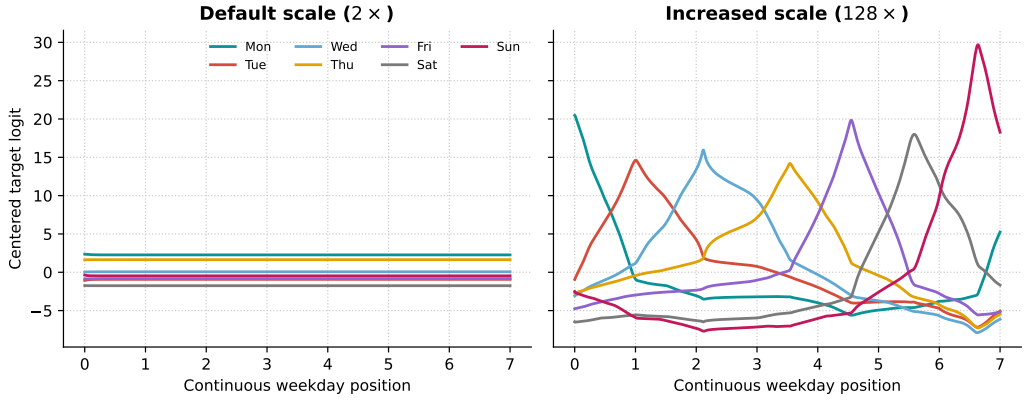


Figure 26: A larger Qwen3-1.7B anchor scale changes nearly constant weekday logits into ordered phase-dependent transitions under recursively composed 0.01-day transformations.

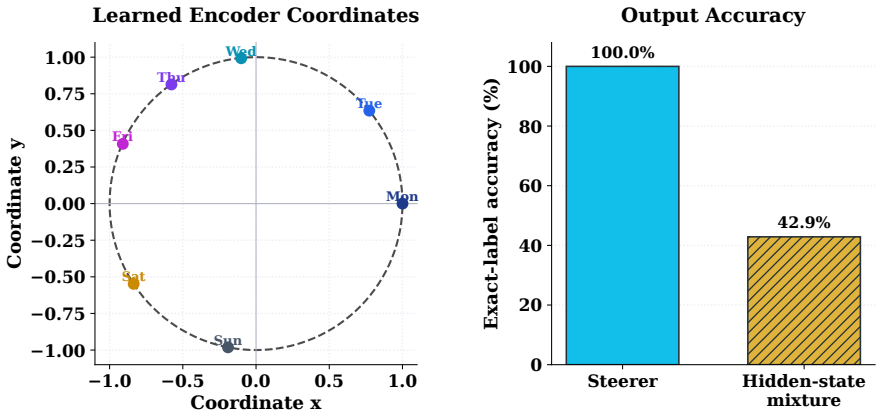


Figure 27: Without coordinate targets, Qwen3-4B encodes the seven hidden-state mixtures in Weekday order. On the seven exact phases, learned decoding makes all labels readable, whereas the direct hidden-state mixture succeeds for only three.

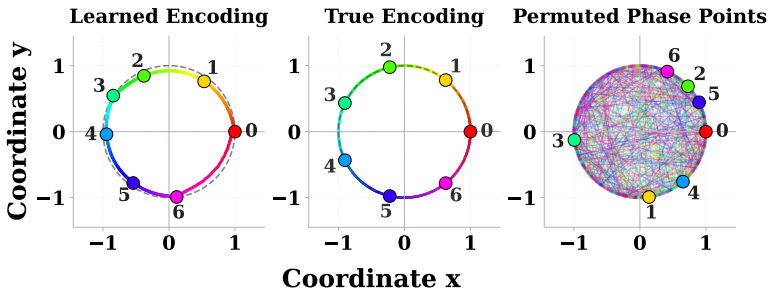


Figure 28: Output supervision organizes Llama3-8B hidden mixtures by circular phase without coordinate regression.

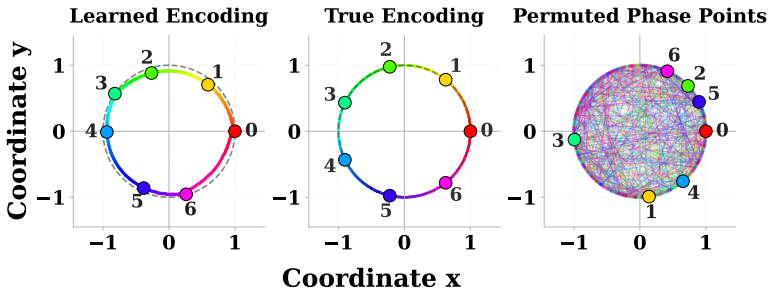


Figure 29: Output supervision organizes Llama3.1-8B hidden mixtures by circular phase without coordinate regression.

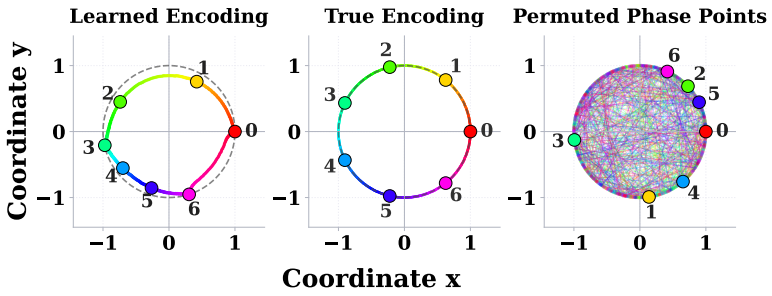


Figure 30: Output supervision organizes Llama3.2-3B hidden mixtures by circular phase without coordinate regression.

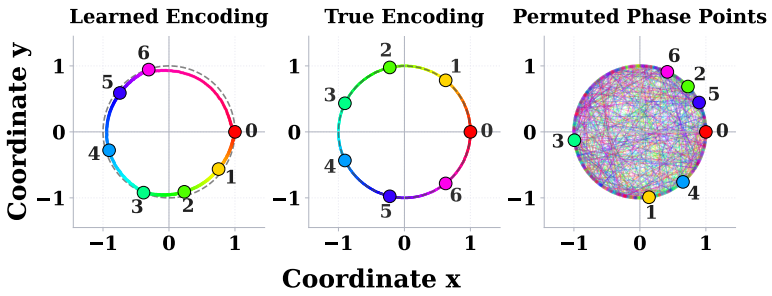


Figure 31: Output supervision organizes Llama3-70B hidden mixtures by circular phase without coordinate regression.

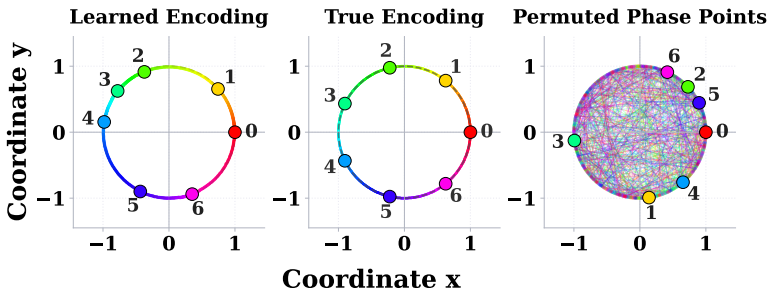


Figure 32: Output supervision organizes Qwen3-1.7B hidden mixtures by circular phase without coordinate regression.

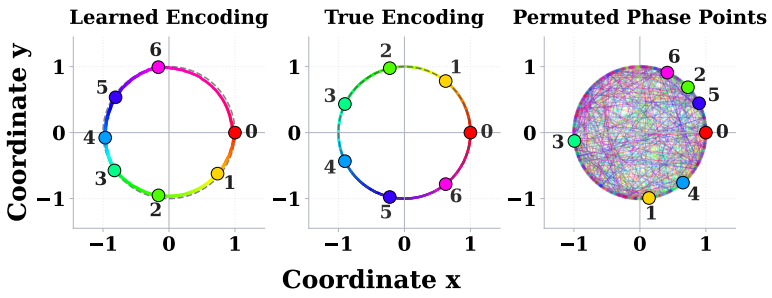


Figure 33: Output supervision organizes Qwen3-8B hidden mixtures by circular phase without coordinate regression.

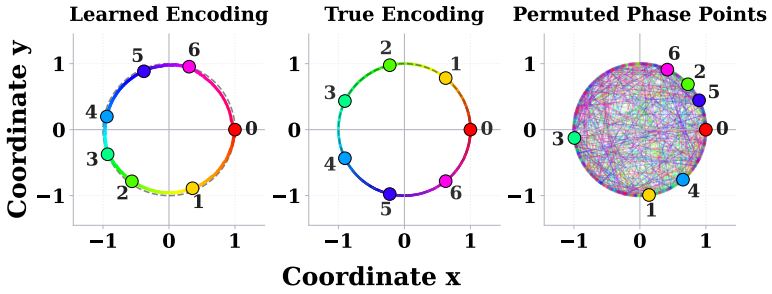


Figure 34: Output supervision organizes Qwen3-14B hidden mixtures by circular phase without coordinate regression.

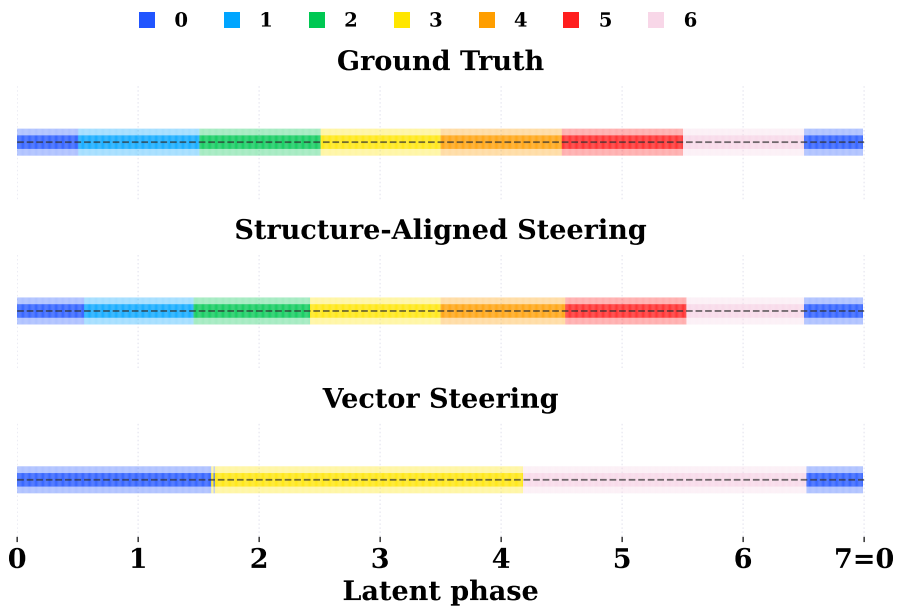


Figure 35: Decoding continuously varying phases produces ordered Qwen3-4B predictions around the complete 0–6–0 cycle.

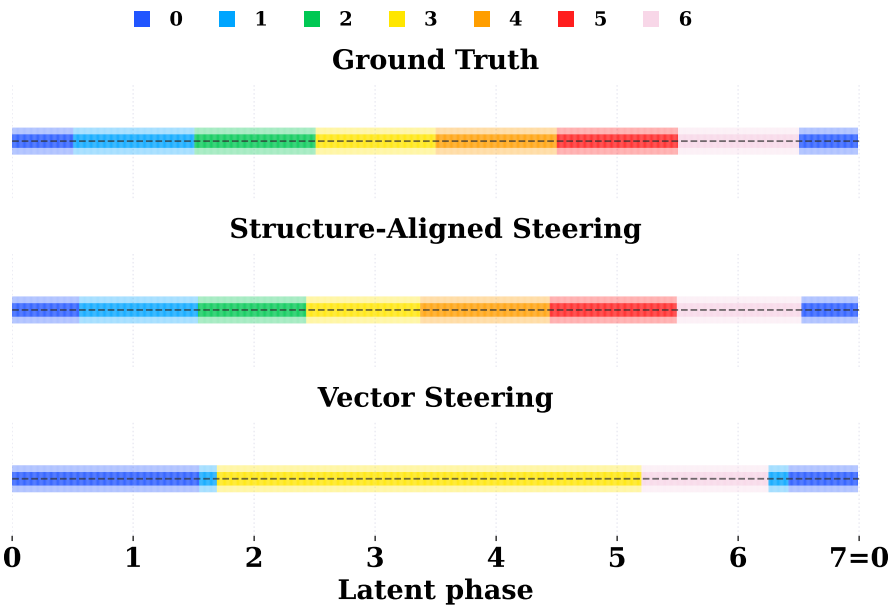


Figure 36: The Llama3-8B decoder produces ordered number predictions around the full latent circle.

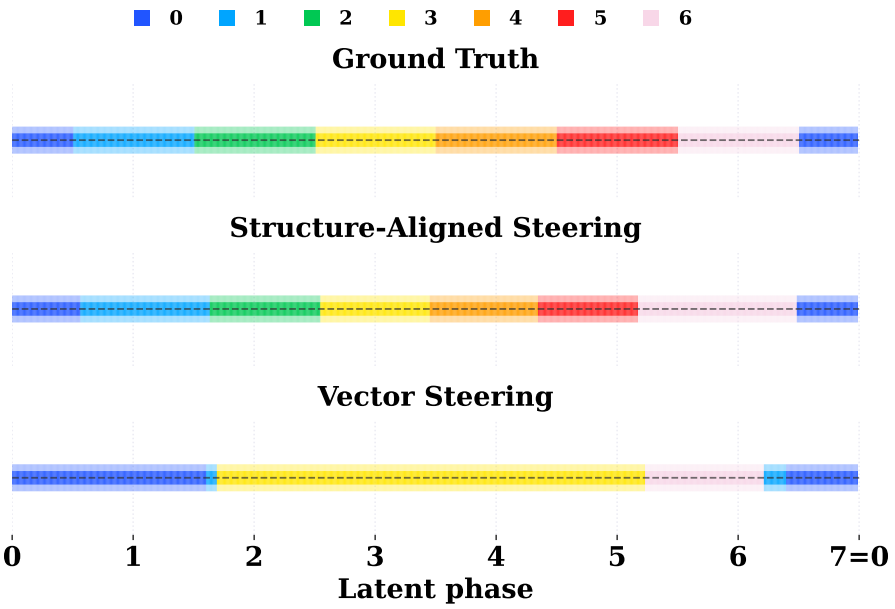


Figure 37: The Llama3.1-8B decoder produces ordered number predictions around the full latent circle.

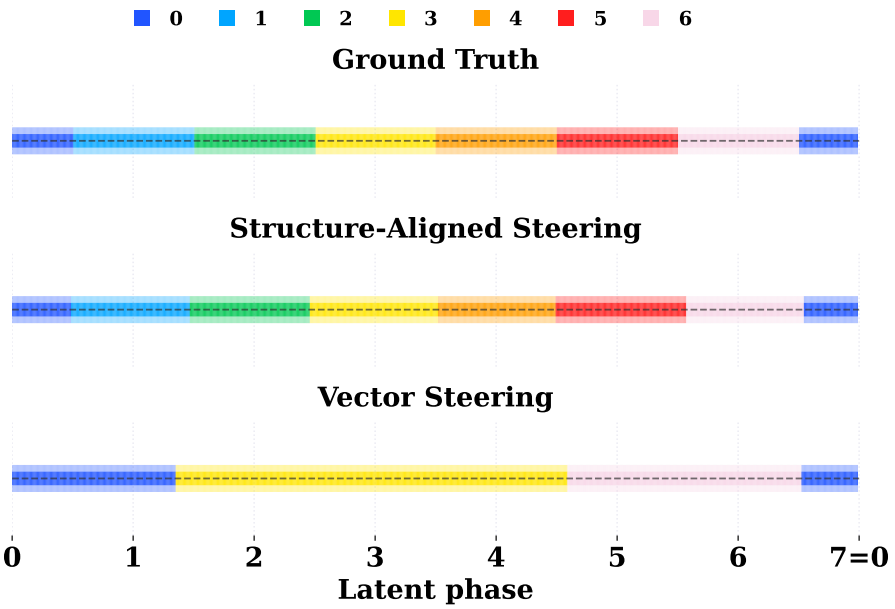


Figure 38: The Llama3.2-3B decoder produces ordered number predictions around the full latent circle.

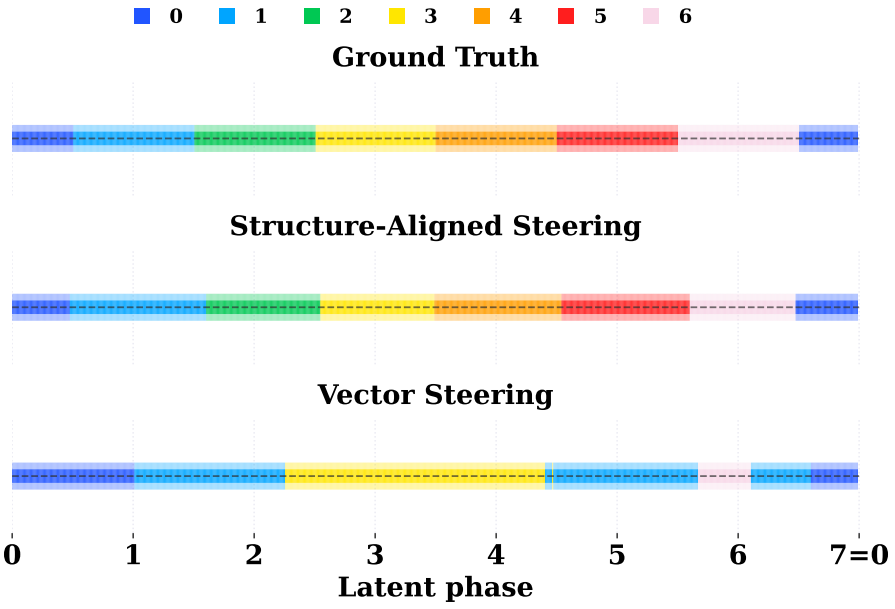


Figure 39: The Llama3-70B decoder produces ordered number predictions around the full latent circle.

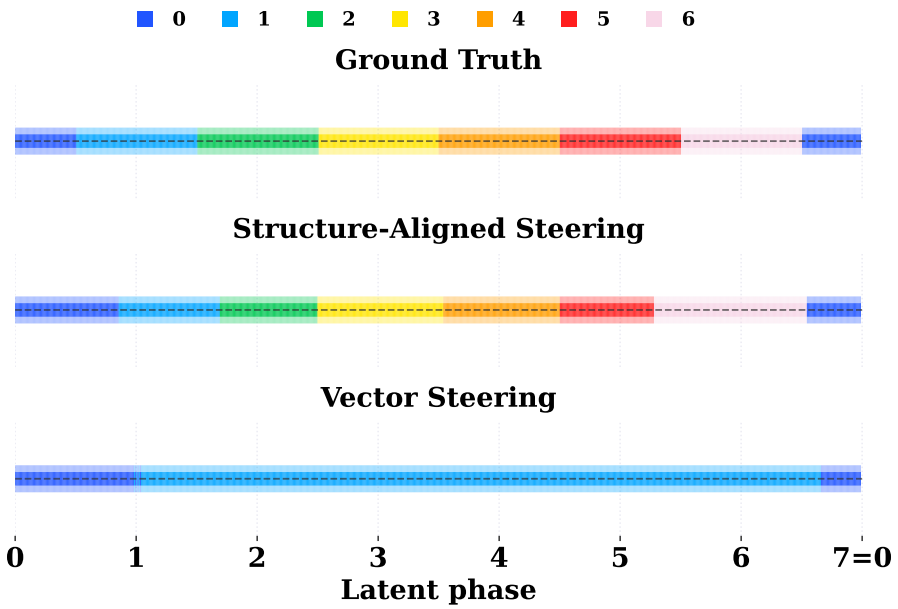


Figure 40: The Qwen3-1.7B decoder produces ordered number predictions around the full latent circle.

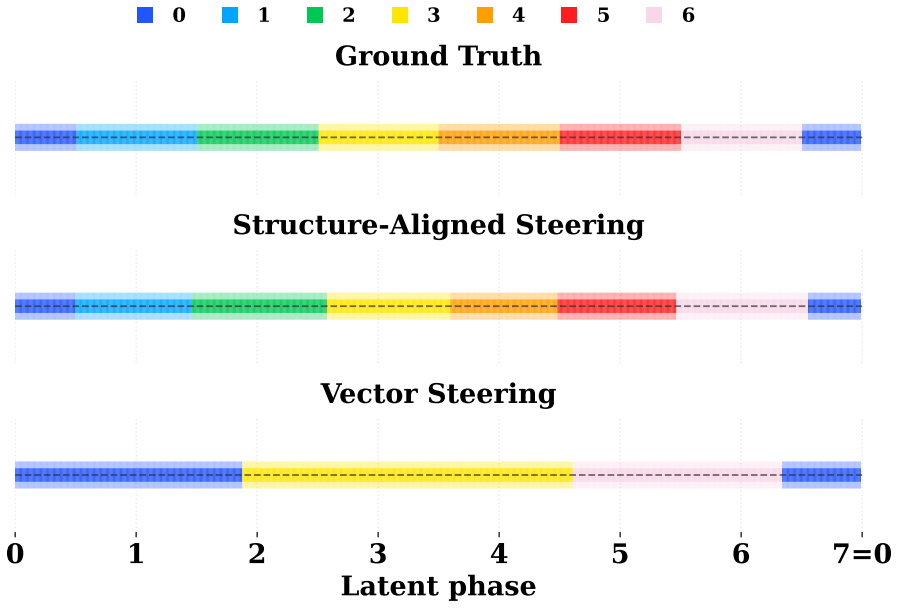


Figure 41: The Qwen3-8B decoder produces ordered number predictions around the full latent circle.

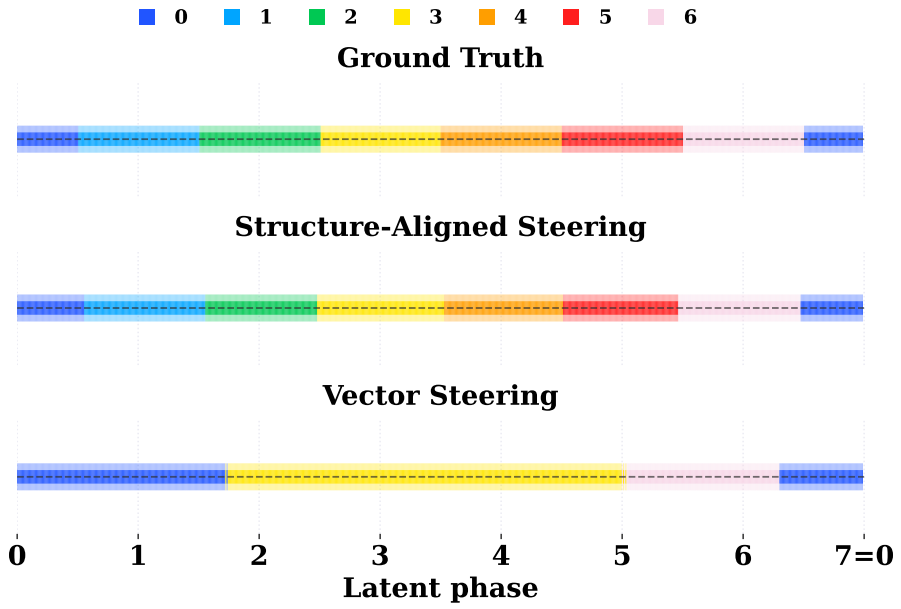


Figure 42: The Qwen3-14B decoder produces ordered number predictions around the full latent circle.

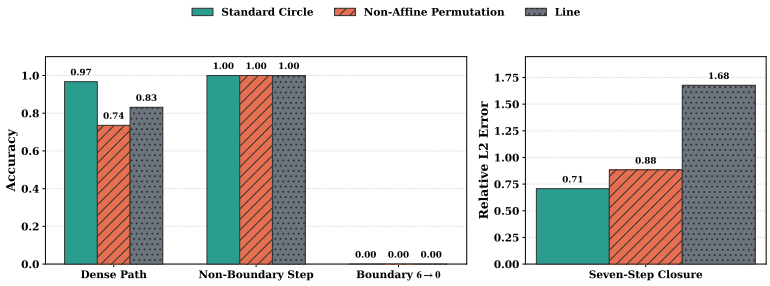


Figure 43: Qwen3-4B achieves strong dense circular decoding but misses the held-out $6 \to 0$ one-step transition.

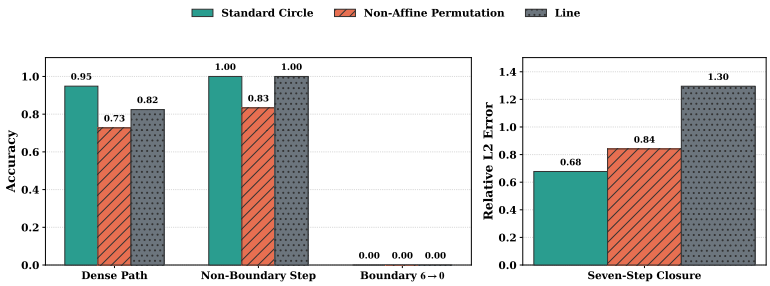


Figure 44: For Llama3-8B, one-step movement, boundary behavior, and full-cycle closure distinguish dense decoding from an exact cyclic action.

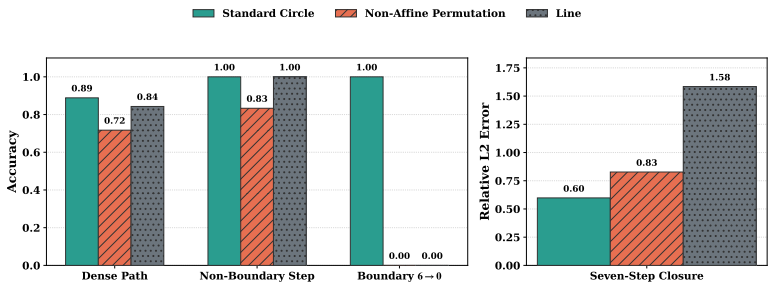


Figure 45: Llama3.1-8B decodes all fitted one-step transitions and the held-out $6 \to 0$ boundary under the task-consistent circle.

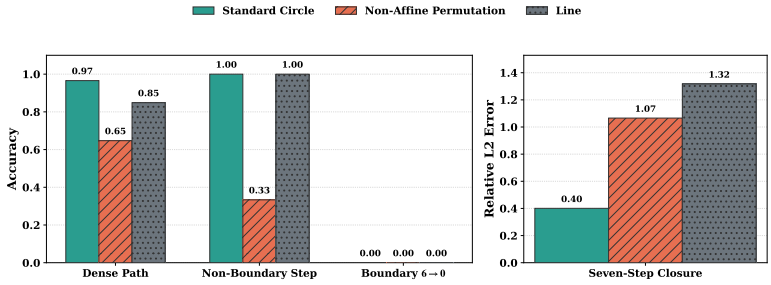


Figure 46: Llama3.2-3B decodes all fitted one-step transitions and the held-out $6 \to 0$ boundary under the task-consistent circle.

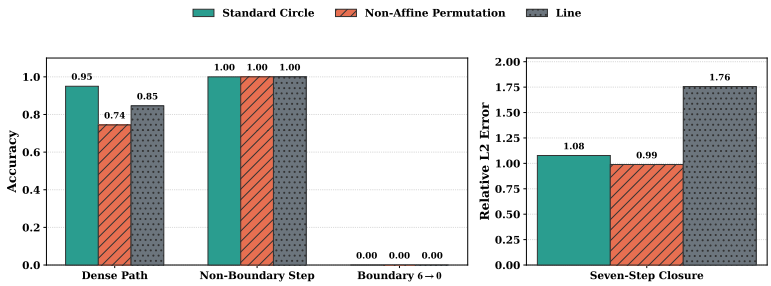


Figure 47: For Llama3-70B, one-step movement, boundary behavior, and full-cycle closure distinguish dense decoding from an exact cyclic action.

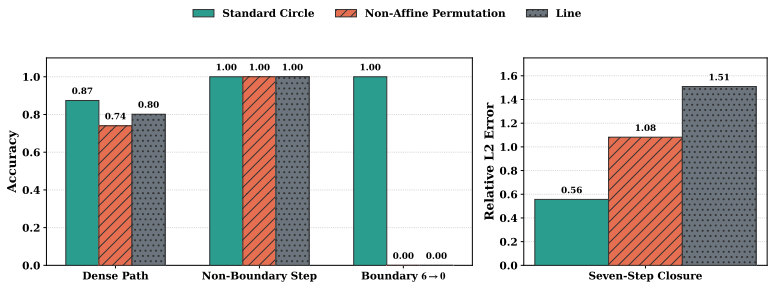


Figure 48: For Qwen3-1.7B, one-step movement, boundary behavior, and full-cycle closure distinguish dense decoding from an exact cyclic action.

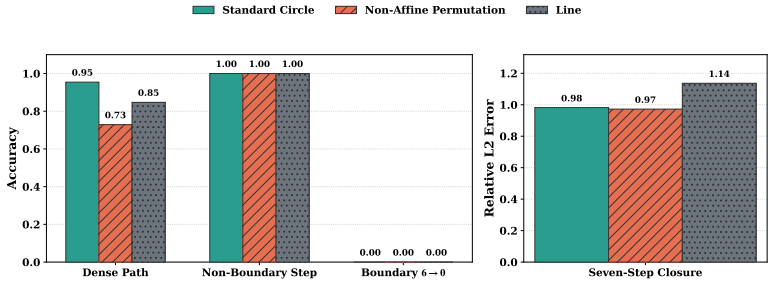


Figure 49: For Qwen3-8B, one-step movement, boundary behavior, and full-cycle closure distinguish dense decoding from an exact cyclic action.

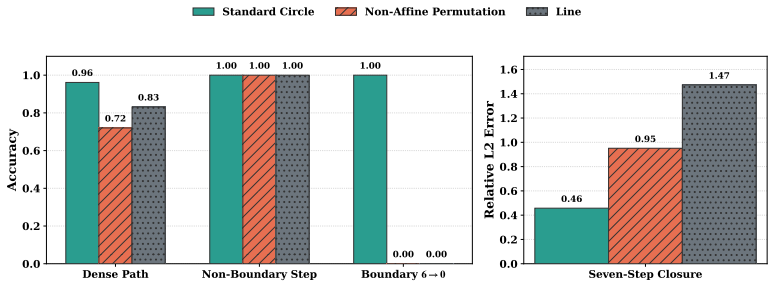


Figure 50: For Qwen3-14B, one-step movement, boundary behavior, and full-cycle closure distinguish dense decoding from an exact cyclic action.

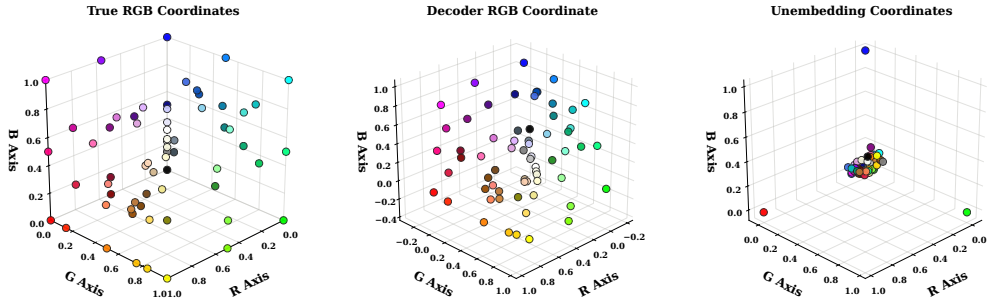


Figure 51: Qwen3-4B color tokens in three coordinate systems: reference RGB, decoded coordinates in the steered red, green, and blue basis, and unembedding coordinates in the basis of the red, green, and blue unembeddings. Only the primaries, which define the unembedding basis, reach their corners; the other colors collapse into one cluster.

Table 17: Top-10 tokens decoded from the leading singular directions of the Qwen3-4B RGB interface at layer 27 (Section F.3). Color tokens are printed in their own hue. Each row shows the sign of the direction that carries them; ␣ marks a leading space and ⟨·⟩ a token this document cannot typeset (Cyrillic, Thai, Hangul, accented Latin). The encoder rows are the six of 30 that decode a color token; the decoder rows are its three leading directions, none of which does.

Direction	Top-10 tokens by unembedding logit
Encoder $+u_9$	`-Core`, `stamp`, `blue`, `dots`, 说是, 但我, ⟨·⟩, `␣Common`, `starter`, 花样
Encoder $-u_{14}$	`green`, `agne`, ⟨·⟩, 逍遥, `\xff`, `TextWriter`, `gesch`, ⟨·⟩, `xffffff`, 仲
Encoder $+u_{23}$	绿色发展, `eness`, 他已经, `"),"`, `ouden`, `green`, `deo`, 跳跃, ⟨·⟩, 可以从
Encoder $-u_{25}$	的一切, `.blue`, `␣BLUE`, `␣Kush`, ⟨·⟩, `-blue`, `pNet`, 蓝色, `/blue`, `finger`
Encoder $-u_{26}$	`ntl`, `␣misdemean`, `unittest`, 单词, `red`, `manship`, 是否, `uably`, `vw`, `NCY`
Encoder $-u_{29}$	`rab`, `subclass`, `erable`, ⟨·⟩, `HT`, `anus`, 沆, `abay`, `/std`, `(rgb`
Decoder $+u_1$	`igram`, `adt`, `␣PARTIC`, `␣CONS`, ⟨·⟩, 湔, `innitus`, `␣CHIP`, `␣inspect`, `geh`
Decoder $+u_2$	`␣america`, `␣verso`, 海岛, `losing`, `egra`, `␣affiliate`, `␣DRAW`, 的商品, `␣Rare`, `␣Supporting`
Decoder $+u_3$	压力, `arness`, 高压, `.mp`, ⟨·⟩, 的压力, `.maven`, `dogs`, `␣mats`, `Diamond`

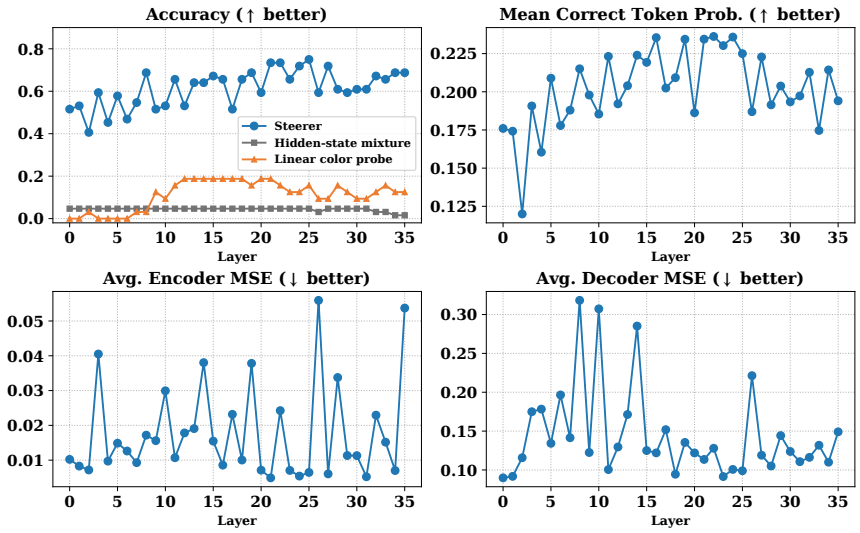


Figure 52: Retraining the Qwen3-4B RGB interface at every layer keeps its accuracy above both the direct hidden-state mixture and the linear color probe fitted at the same layer, and keeps encoder MSE low at most layers.

Table 18: RGB steerers trained on the task-consistent geometry, a distance-scale-matched cubic warp, or two- and one-dimensional PCA projections. Continuous accuracy (%) is measured over sampled coordinates; distance correlation compares encoded and true RGB distances.

	Continuous accuracy				Distance correlation			
Model	3D	Warp	2D	1D	3D	Warp	2D	1D
Llama3-8B	**68.9**	31.7	29.5	6.2	**0.896**	0.715	0.828	0.581
Llama3.1-8B	**69.6**	23.8	28.9	5.7	**0.759**	0.668	0.705	0.582
Llama3.2-3B	**57.4**	32.4	30.4	6.4	**0.908**	0.749	0.719	0.563
Llama3-70B	**62.0**	31.0	28.0	6.9	**0.859**	0.755	0.716	0.582
Qwen3-1.7B	**27.3**	6.4	7.1	4.7	**0.724**	0.288	0.450	0.560
Qwen3-4B	**62.7**	26.7	27.8	6.5	**0.808**	0.752	0.702	0.579
Qwen3-8B	**62.6**	18.6	27.1	6.9	0.737	0.581	**0.762**	0.559
Qwen3-14B	**60.0**	8.8	22.2	7.9	**0.709**	0.311	0.643	0.574

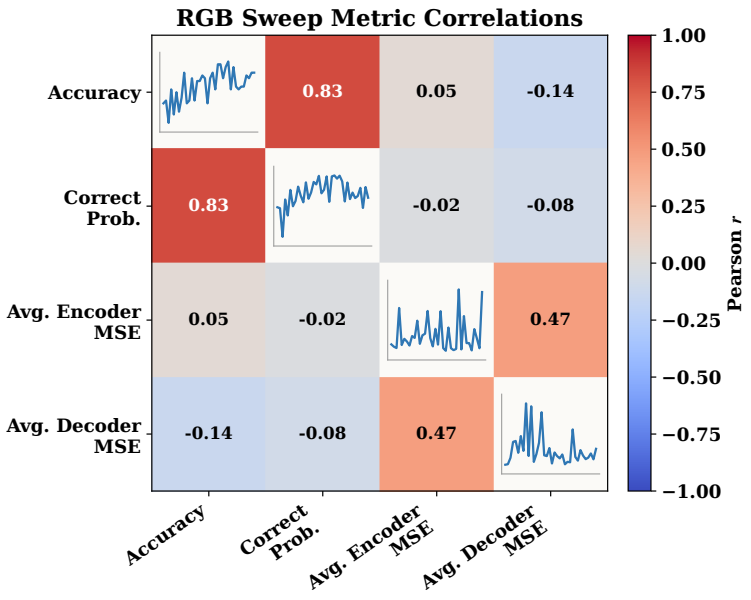


Figure 53: Across Qwen3-4B layers, accuracy tracks the mean correct-token probability ($r = 0.83$) but is nearly uncorrelated with encoder and decoder MSE, separating remaining-layer readability from learned geometric organization.

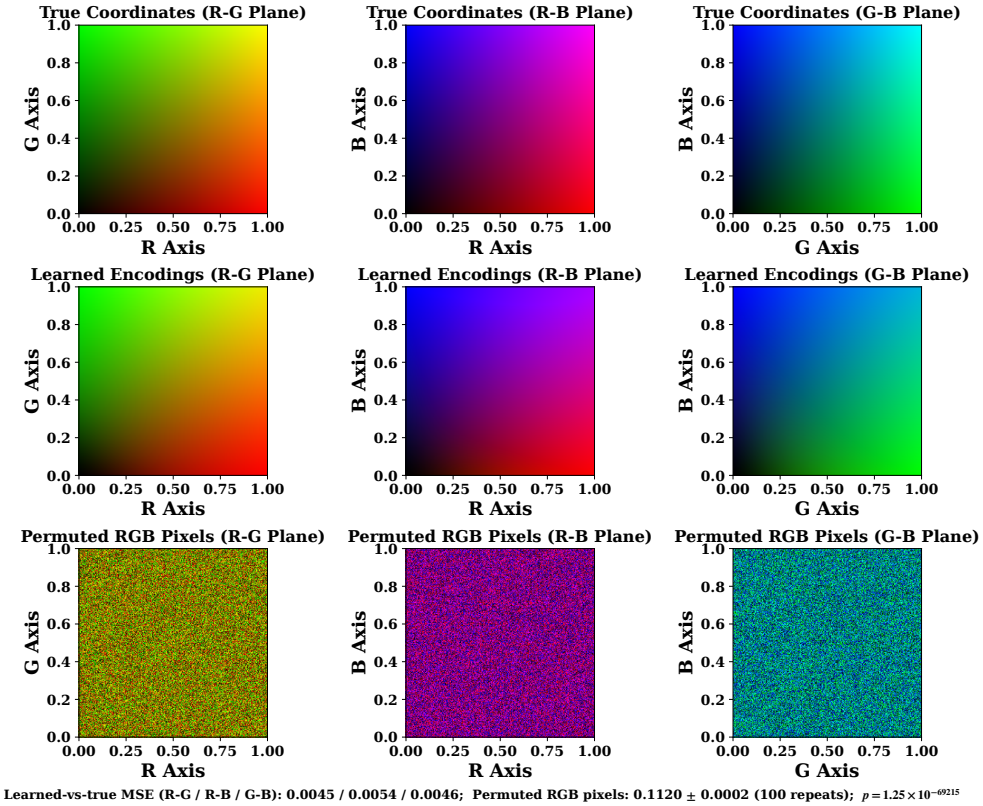


Figure 54: After four-point alignment, the Llama3-8B encoder reconstructs the channel-wise organization of three RGB cube faces, unlike permuted pixels.

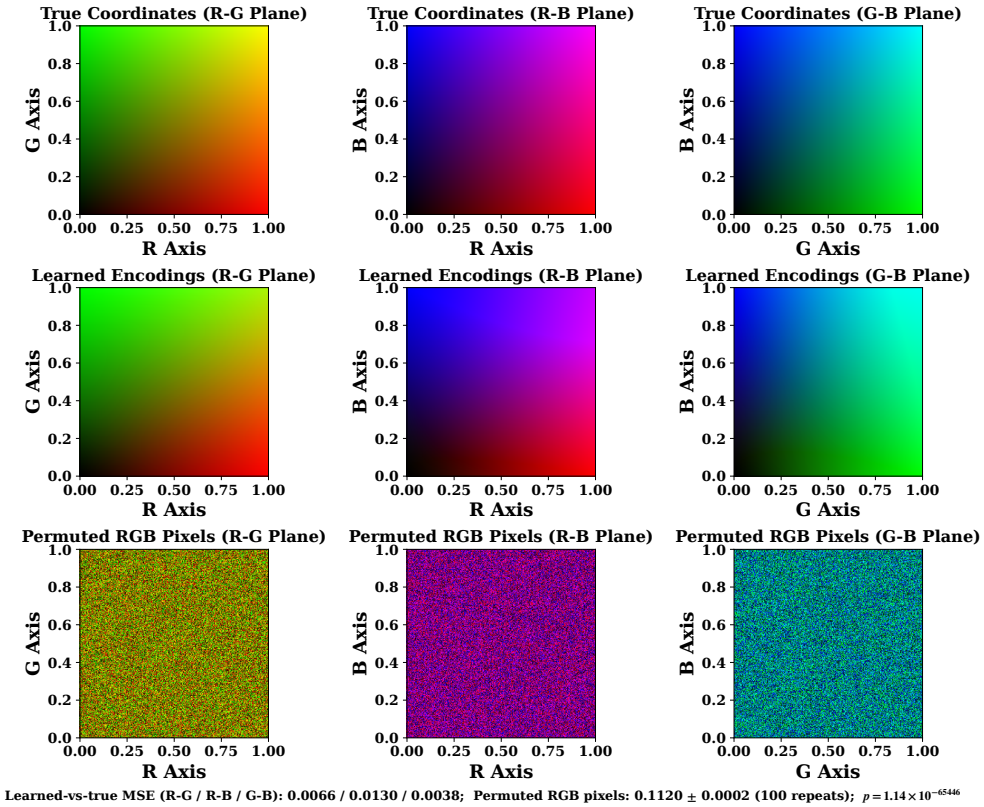


Figure 55: After four-point alignment, the Llama3.1-8B encoder reconstructs the channel-wise organization of three RGB cube faces, unlike permuted pixels.

Table 19: Continuous accuracy (%) of the task-consistent (True) and cubic-warp (Warp) RGB steerers, each scored against the nearest color in its own training geometry rather than in true RGB.

Model	True	Warp
Llama3-8B	**68.9**	54.9
Llama3.1-8B	**69.6**	45.9
Llama3.2-3B	**57.4**	53.7
Llama3-70B	**62.0**	48.4
Qwen3-1.7B	**27.3**	6.4
Qwen3-4B	**62.7**	50.2
Qwen3-8B	**62.6**	28.9
Qwen3-14B	**60.0**	11.6

Table 20: The layer-local LoRA and Prefix baselines meet or exceed the $840d$ adaptation budget for every model, while the complete Structure-Aligned solver remains below $840d$.

Model	Ours params	Ours/d	LoRA r	LoRA params	LoRA/d	Prefix tokens	Prefix params	Prefix/d
Llama3-8B	3,290,942	803.45	42	3,440,640	840.00	1,680	3,440,640	840.00
Llama3.1-8B	3,290,942	803.45	42	3,440,640	840.00	1,680	3,440,640	840.00
Llama3.2-3B	2,503,486	814.94	48	2,605,056	848.00	1,260	2,580,480	840.00
Llama3-70B	6,440,766	786.23	43	6,957,056	849.25	3,360	6,881,280	840.00
Qwen3-1.7B	1,716,030	837.91	45	1,751,040	855.00	840	1,720,320	840.00
Qwen3-4B	2,109,758	824.12	38	2,179,072	851.20	1,050	2,150,400	840.00
Qwen3-8B	3,290,942	803.45	46	3,485,696	851.00	1,680	3,440,640	840.00
Qwen3-14B	4,078,398	796.56	43	4,315,136	842.80	2,100	4,300,800	840.00

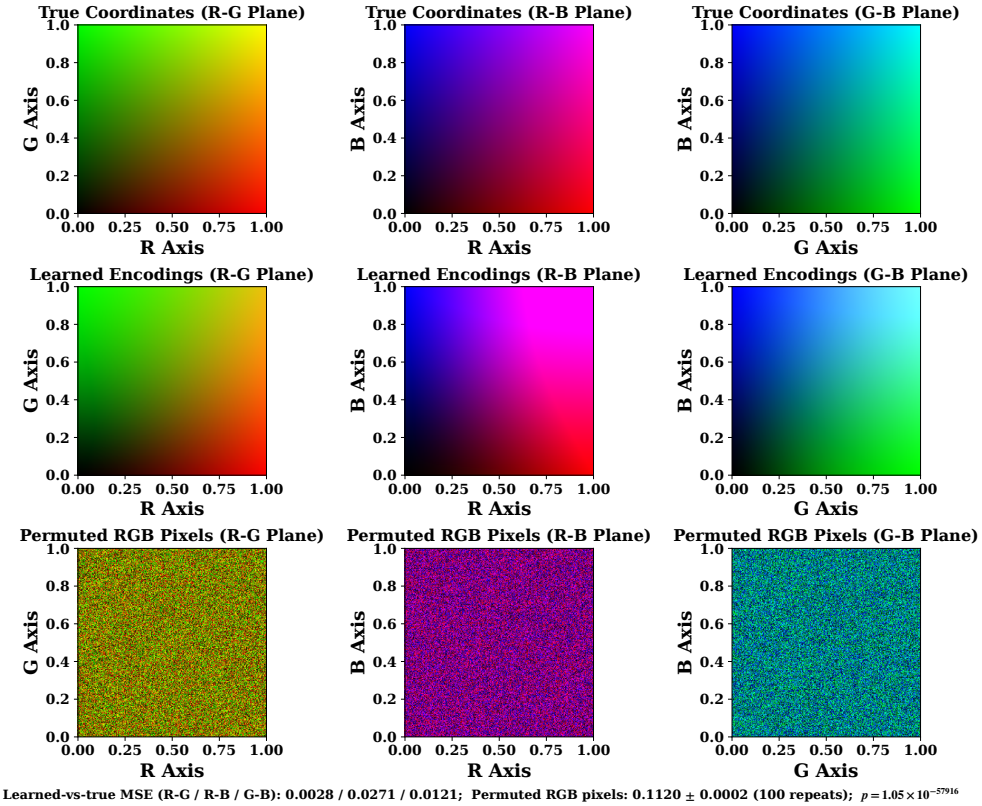


Figure 56: After four-point alignment, the Llama3.2-3B encoder reconstructs the channel-wise organization of three RGB cube faces, unlike permuted pixels.

Table 21: Structure-Aligned Steering outperforms the similarly sized PEFT methods and logistic baseline on pair-disjoint modular queries across models. Structure-Aligned and oracle-decoder values use the canonical deep per-model artifacts; comparison methods use the matched 18,000-example runs.

Model	Structure-Aligned	Logistic	LoRA	Prefix	Oracle Decoder
Llama3-8B	**.982**	.256	.766	.148	1.000
Llama3.1-8B	**.988**	.262	.716	.170	1.000
Llama3.2-3B	**.962**	.010	.618	.180	1.000
Llama3-70B	**.996**	.078	.460	.000	1.000
Qwen3-1.7B	**.938**	.034	.238	.060	1.000
Qwen3-4B	**.872**	.070	.342	.088	1.000
Qwen3-8B	**.958**	.070	.366	.012	1.000
Qwen3-14B	**.948**	.054	.346	.140	1.000

Table 22: Qwen3-4B zero-shot modular accuracy when the same aligner predictions are written at layer 27 either as the steerer's decoded residue state or as its residue anchor. Predicted and true rows use the aligner's residue and the correct residue, respectively; the unmodified-model row repeats the baseline of Figure 8.

Written state	Predicted residue	True residue
Decoded state $D_\phi(E_\theta(a_r))$	.872	1.000
Residue anchor a_r	.872	1.000
Aligner result-residue accuracy	.872	–
Unmodified model	.246	

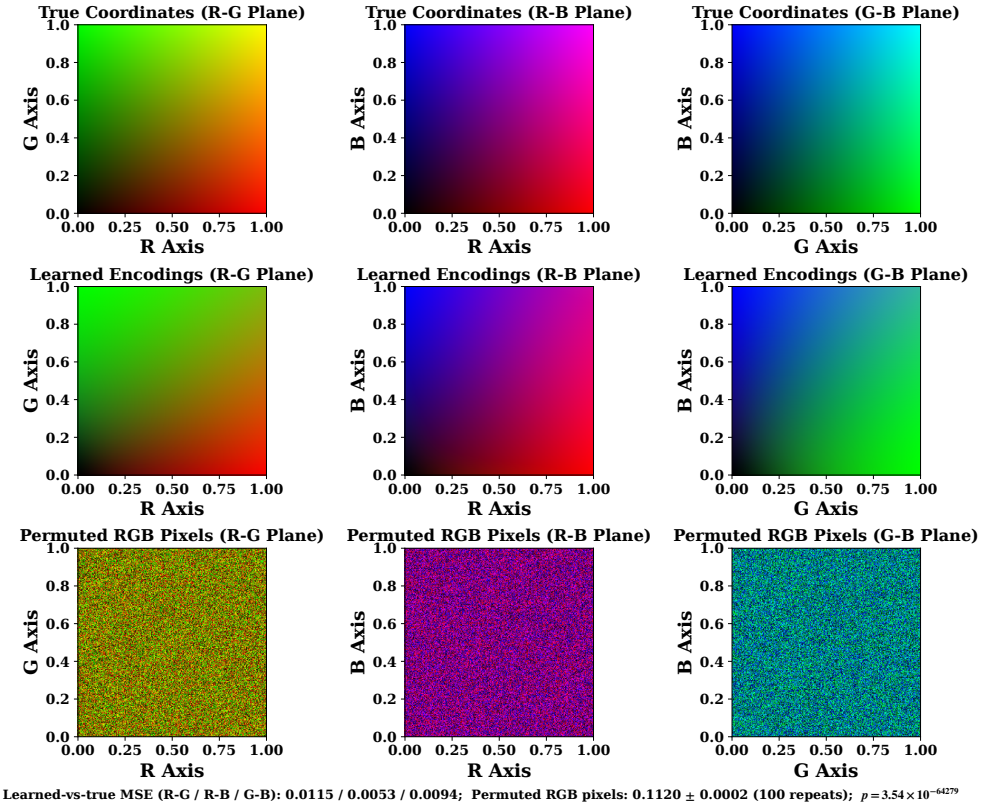


Figure 57: After four-point alignment, the Llama3-70B encoder reconstructs the channel-wise organization of three RGB cube faces, unlike permuted pixels.

Table 23: Zero-shot query templates used for modular prompt-transfer evaluation. Braced fields are replaced by integer operands and the modulus; every query targets $r = (a + b) \bmod m$.

Template	Query prefix
Standard	`{a}+{b} mod {m}, the answer is`
Worded	`What is ({a} plus {b}) modulo {m}? Answer:`
Remainder	`The remainder when {a}+{b} is divided by {m} is`
Function-style	`mod({a}+{b}, {m}) =`
Instruction-style	`Compute {a}+{b} in Z_{m}. Result:`

Table 24: Modular accuracy (%) under the original template and four new templates without retraining, on 500 prompts from optimization pairs (Opt.) and from held-out test pairs (Test). Templates are listed in Table 23.

	Original		Words		Remainder		Function		Instruction	
Model	Opt.	Test	Opt.	Test	Opt.	Test	Opt.	Test	Opt.	Test
Llama3-8B	100.0	98.4	50.2	38.0	48.0	46.4	60.2	52.2	51.8	41.4
Llama3.1-8B	100.0	99.0	49.6	40.2	21.6	19.2	56.8	38.2	60.0	45.6
Llama3.2-3B	99.4	95.8	28.0	26.8	47.2	43.6	38.4	25.8	23.6	19.2
Llama3-70B	100.0	99.4	24.2	20.0	23.6	18.6	21.0	21.2	24.2	18.8
Qwen3-1.7B	99.2	93.2	30.6	31.4	25.4	21.2	30.8	30.4	24.0	28.2
Qwen3-4B	99.2	87.4	44.8	38.4	23.0	22.8	22.6	15.2	21.8	21.0
Qwen3-8B	100.0	96.0	27.6	22.4	19.6	11.6	28.8	26.2	28.8	29.0
Qwen3-14B	99.8	94.8	32.8	33.8	42.8	43.4	46.2	44.4	40.2	39.0

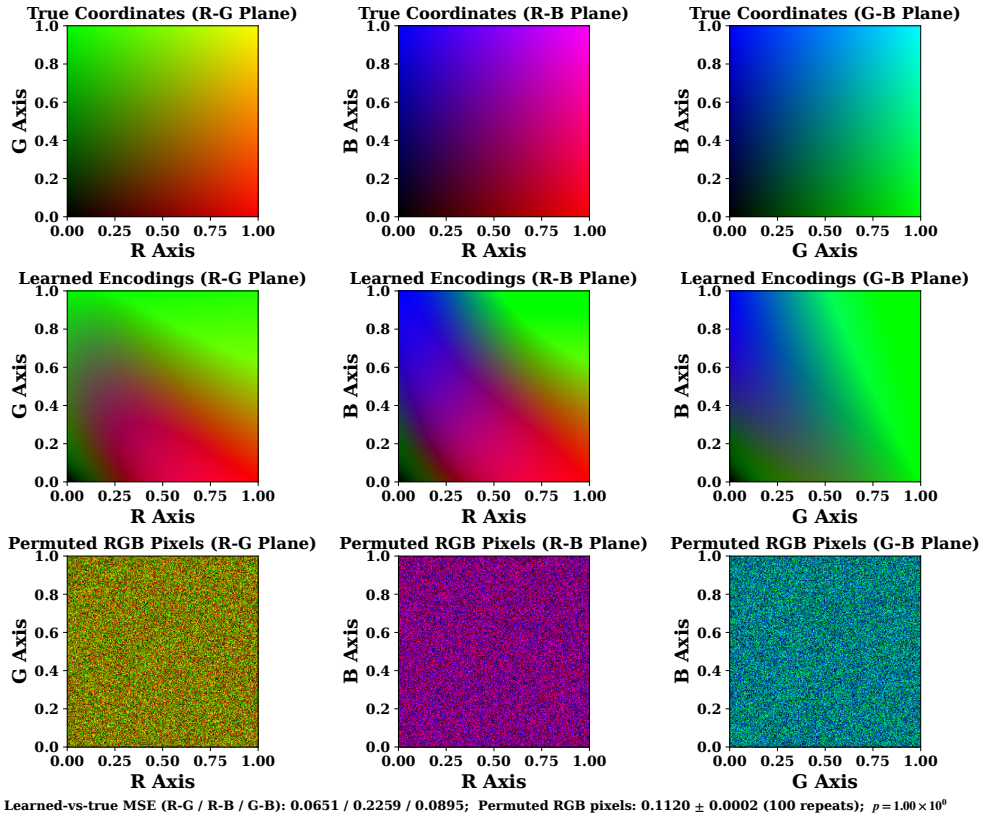


Figure 58: After four-point alignment, the Qwen3-1.7B encoder reconstructs the channel-wise organization of three RGB cube faces, unlike permuted pixels.

Table 25: On Qwen3-4B, an additional $\mathbb{Z}_7$ action preserves modular accuracy under both prompt settings; the oracle starts from the true result coordinate.

Prompt	Original	After $\mathbb{Z}_7$	Oracle
Zero-shot	.872	.872	1.000
16-shot	.822	.822	1.000

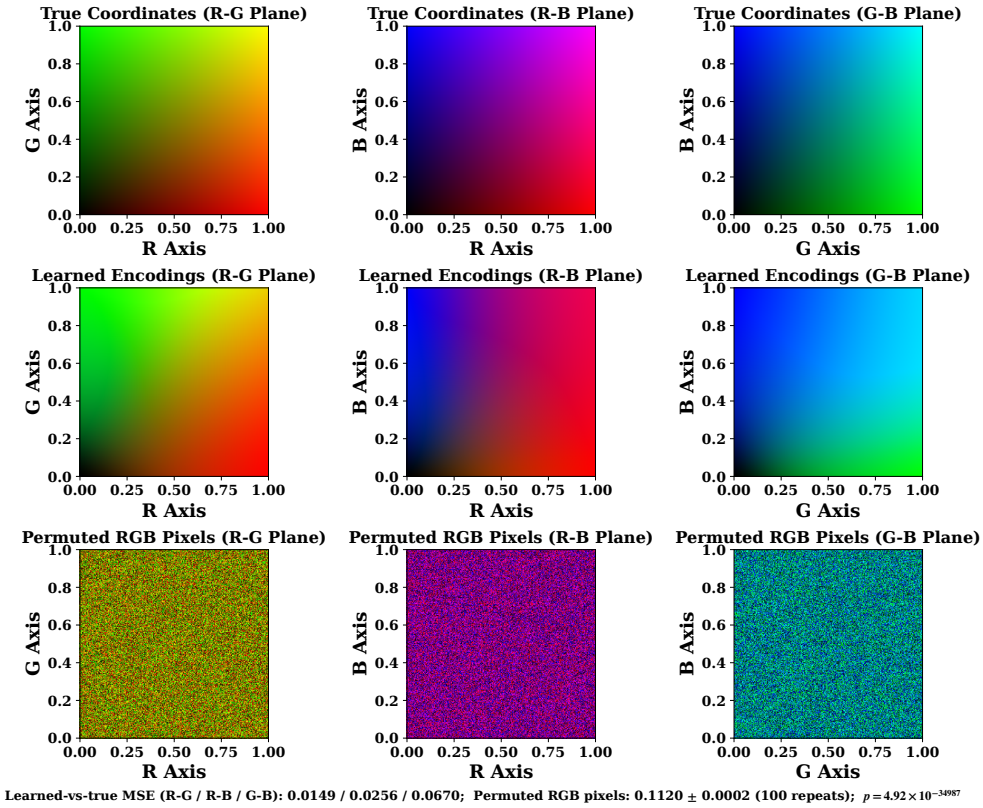


Figure 59: After four-point alignment, the Qwen3-8B encoder reconstructs the channel-wise organization of three RGB cube faces, unlike permuted pixels.

Table 26: Operand-token intervention on Llama-3.1-8B, over 16 carriers and all seven target weekdays. Answer accuracy is the fraction of carrier–day pairs whose intervened state produces the required arithmetic answer; base-solvable accuracy restricts this metric to pairs answered correctly by the unmodified model on the matched natural prompt. $\Delta \log p$ is the aggregated weekday log-probability difference between the target and original answers, and phase accuracy scores fractional-phase outputs. Discussion is in Section I.2.

Layer	Method	Answer accuracy	Answer accuracy (base-solvable)	$\Delta \log p$	Phase accuracy
–	Unmodified LLM	.929	–	–	–
4	Ours	**.964**	**1.000**	**+4.05**	**.879**
	Engels et al.	.768	.788	+3.48	.667
8	Ours	**.955**	**1.000**	**+4.02**	**.867**
	Engels et al.	.759	.779	+3.46	.677
12	Ours	**.964**	**1.000**	**+4.00**	**.861**
	Engels et al.	.732	.750	+3.36	.664
16	Ours	**.964**	**1.000**	**+3.83**	**.850**
	Engels et al.	.786	.808	+3.24	.671
20	Ours	.214	.231	−2.83	.194
	Engels et al.	.161	.173	−2.96	.158
24	Ours	.161	.173	−4.05	.150
	Engels et al.	.143	.154	−4.13	.144

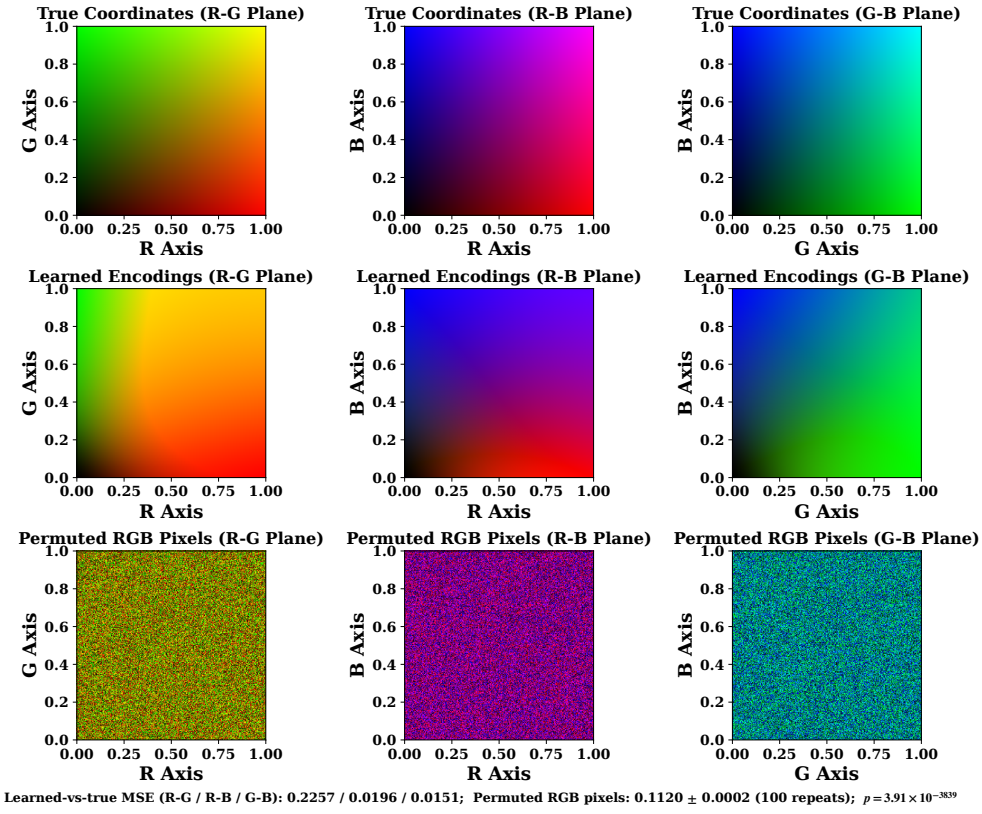


Figure 60: After four-point alignment, the Qwen3-14B encoder reconstructs the channel-wise organization of three RGB cube faces, unlike permuted pixels.

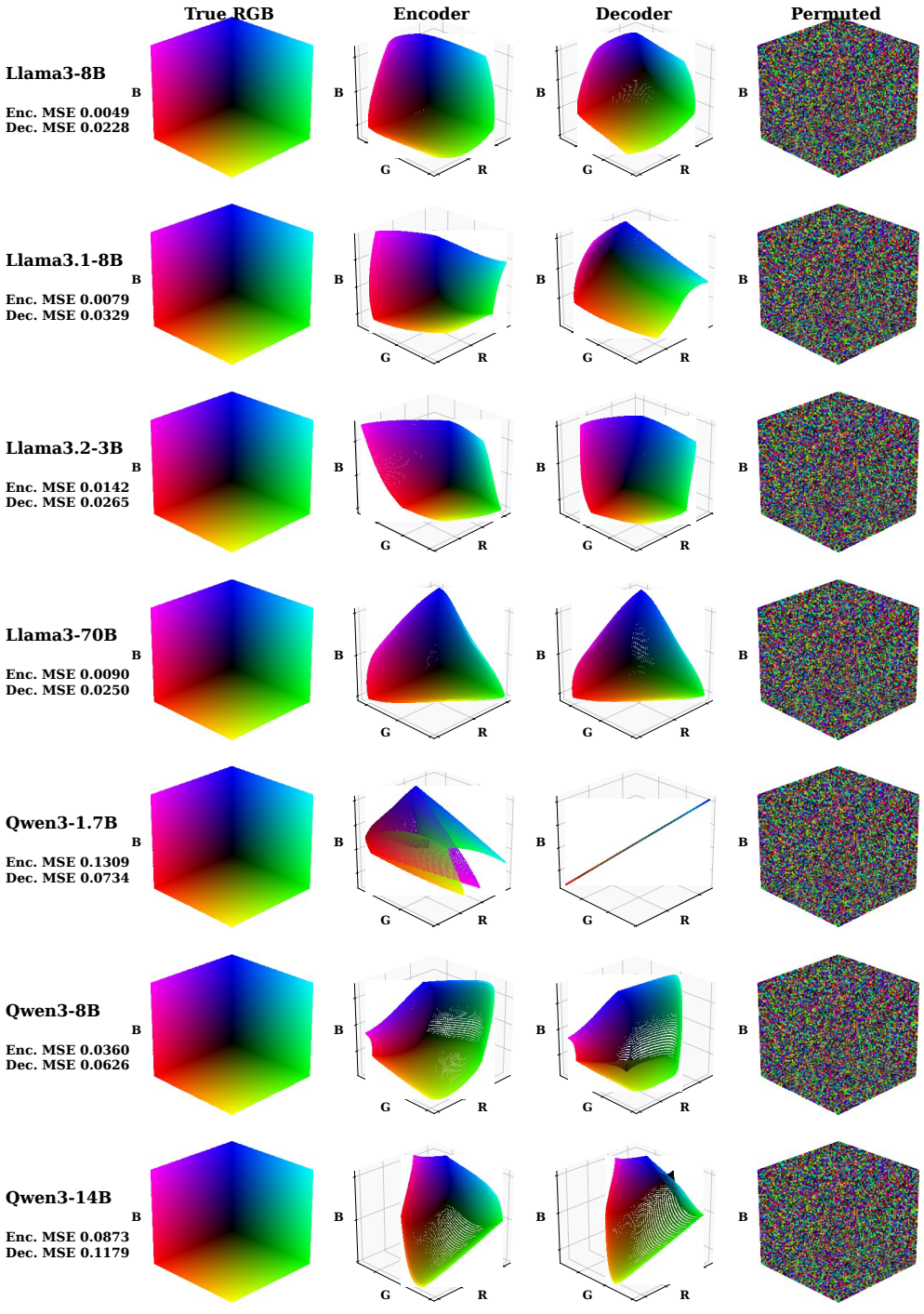


Figure 61: Encoder and decoder geometry for the seven additional models, drawn as in Figure 5. Each row shows the reference coordinates, the aligned encodings, the decoded displacements in the steered red, green, and blue basis, and a permutation of the same points, whose MSE is 0.2256 ± 0.0013 for every model. The numbers under each model name are the MSE between the true RGB coordinates and that model's aligned encoder (Enc.) or decoder (Dec.) coordinates.

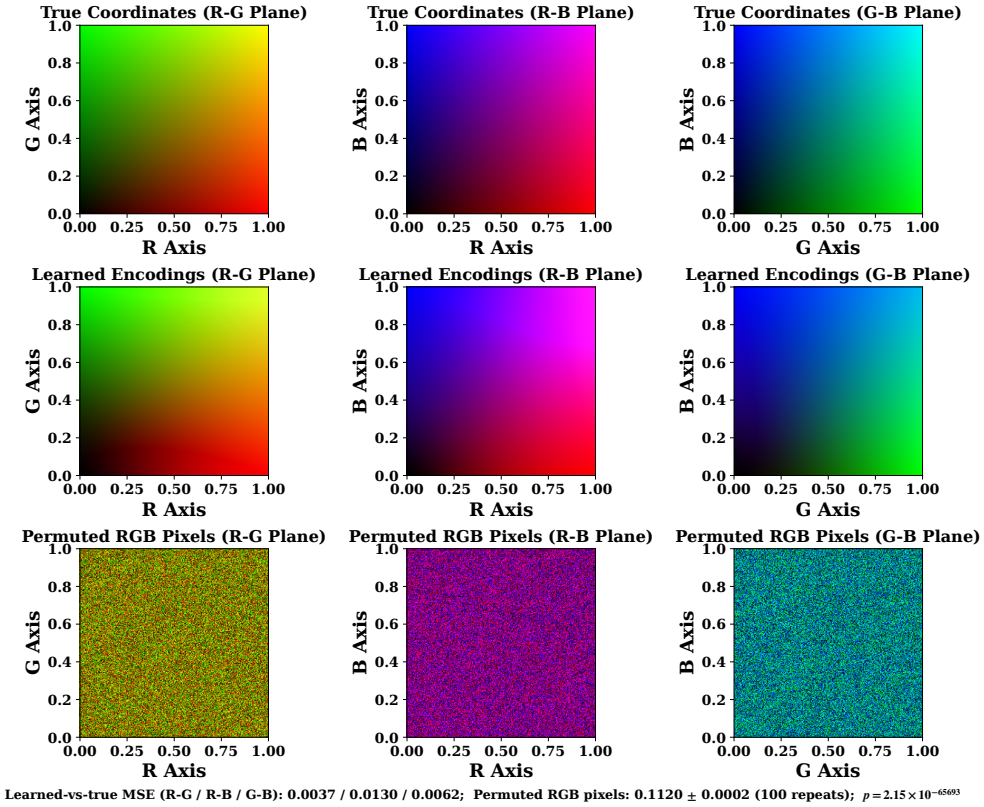


Figure 62: Viewed one face at a time, the Qwen3-4B encoder reconstructs the channel-wise organization of three RGB cube faces (MSE 0.0076), unlike permuted pixels (MSE 0.1120).

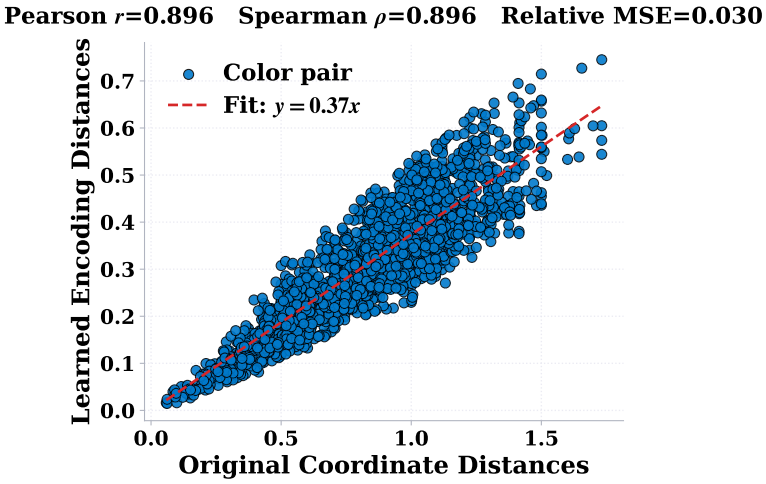


Figure 63: On Llama3-8B, pairwise encoded distances track RGB-space distances across the complete reference set after one global rescaling.

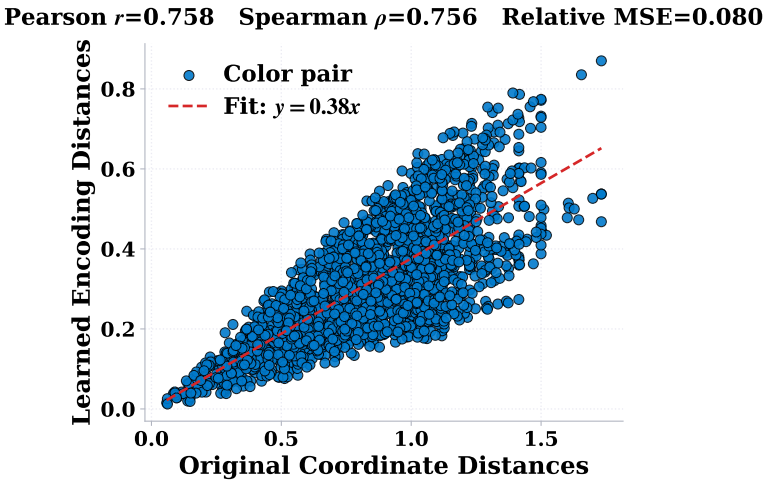


Figure 64: On Llama3.1-8B, pairwise encoded distances track RGB-space distances across the complete reference set after one global rescaling.

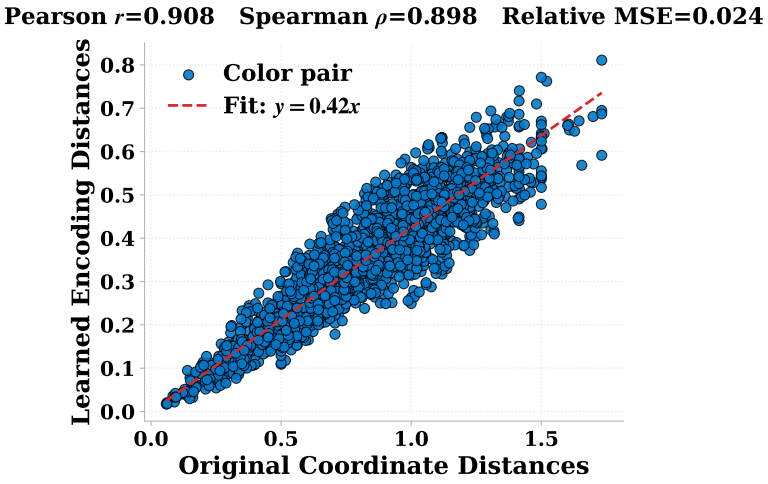


Figure 65: On Llama3.2-3B, pairwise encoded distances track RGB-space distances across the complete reference set after one global rescaling.

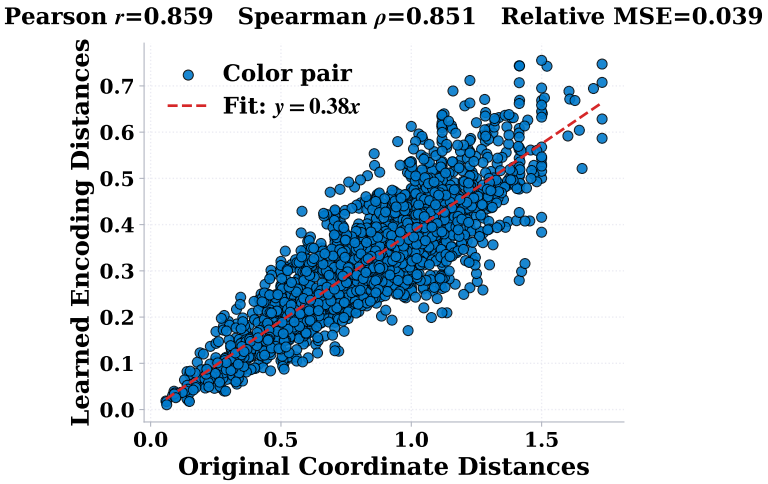


Figure 66: On Llama3-70B, pairwise encoded distances track RGB-space distances across the complete reference set after one global rescaling.

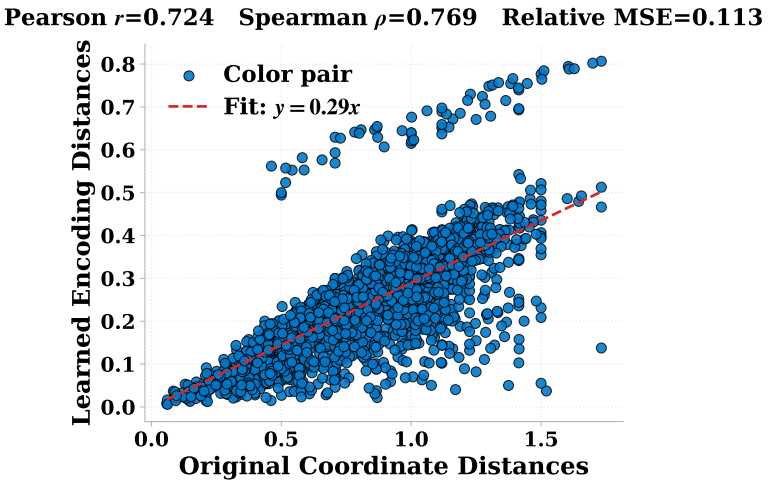


Figure 67: On Qwen3-1.7B, pairwise encoded distances track RGB-space distances across the complete reference set after one global rescaling.

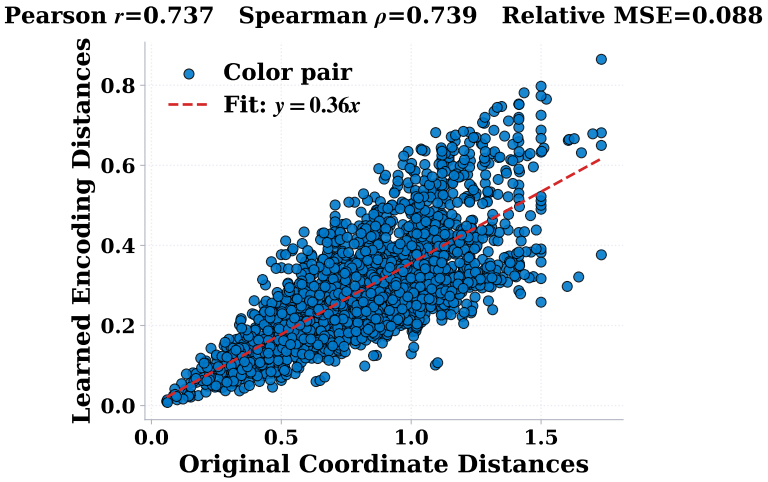


Figure 68: On Qwen3-8B, pairwise encoded distances track RGB-space distances across the complete reference set after one global rescaling.

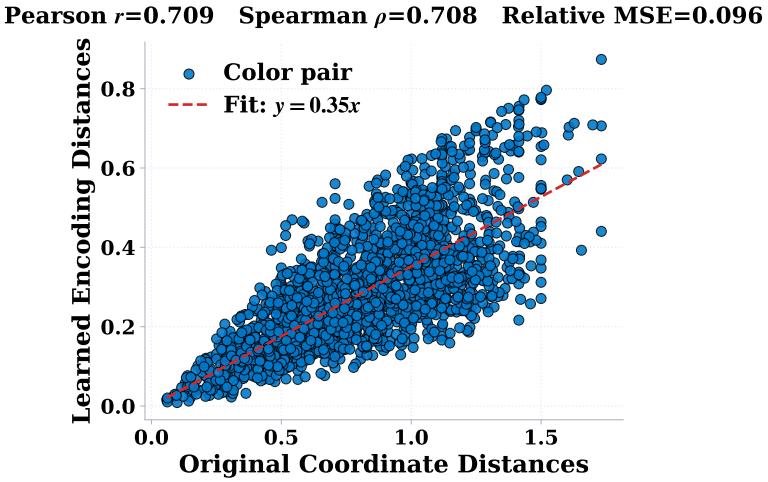


Figure 69: On Qwen3-14B, pairwise encoded distances track RGB-space distances across the complete reference set after one global rescaling.

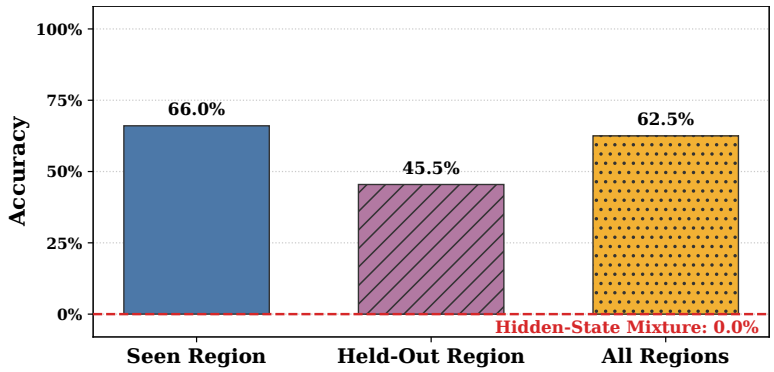


Figure 70: With a contiguous yellow–orange region excluded from training, Qwen3-4B retains partial accuracy inside it while direct hidden-state mixtures fail.

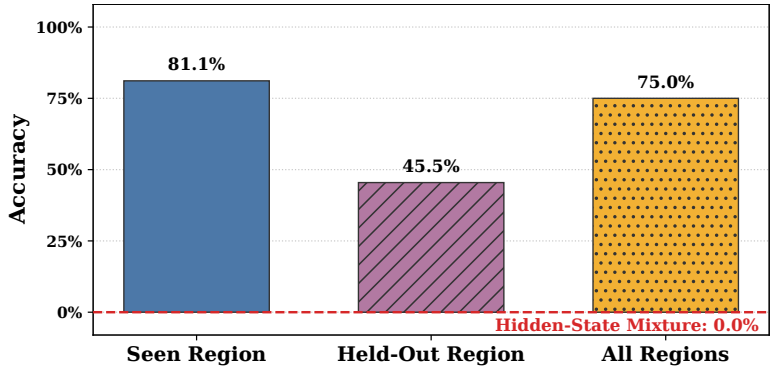


Figure 71: Llama3-8B retains partial RGB accuracy in a contiguous region omitted from training, with a gap from seen-region performance.

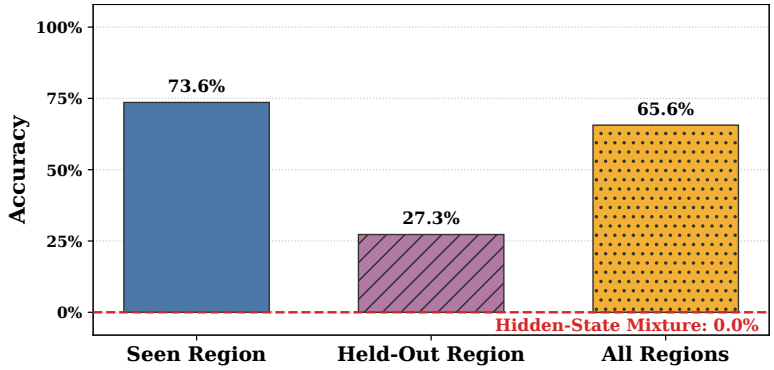


Figure 72: Llama3.1-8B retains partial RGB accuracy in a contiguous region omitted from training, with a gap from seen-region performance.

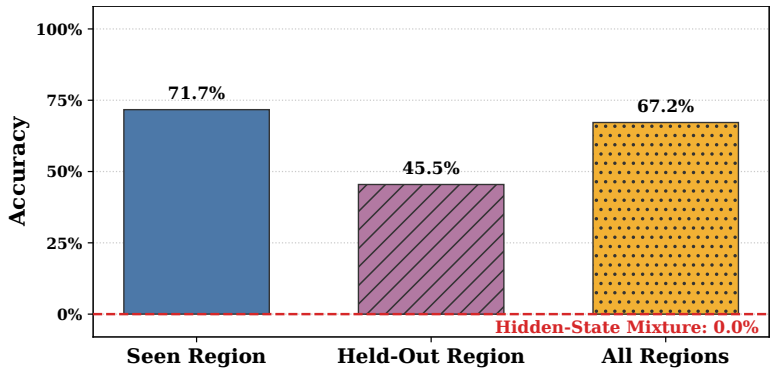


Figure 73: Llama3.2-3B retains partial RGB accuracy in a contiguous region omitted from training, with a gap from seen-region performance.

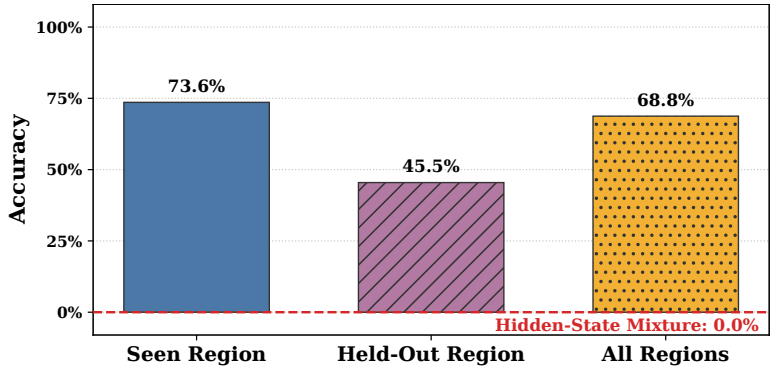


Figure 74: Llama3-70B retains partial RGB accuracy in a contiguous region omitted from training, with a gap from seen-region performance.

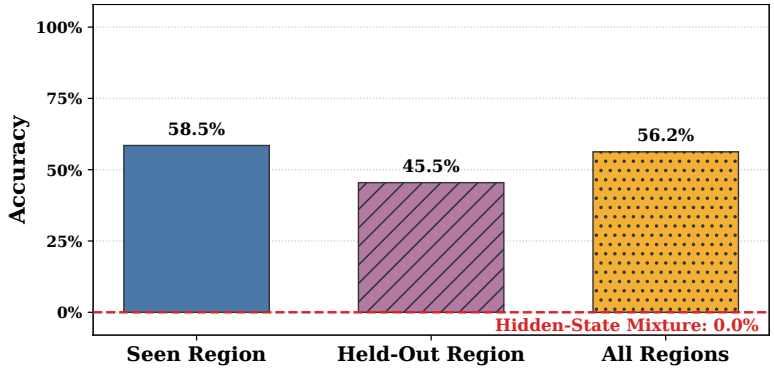


Figure 75: Qwen3-1.7B retains partial RGB accuracy in a contiguous region omitted from training, with a gap from seen-region performance.

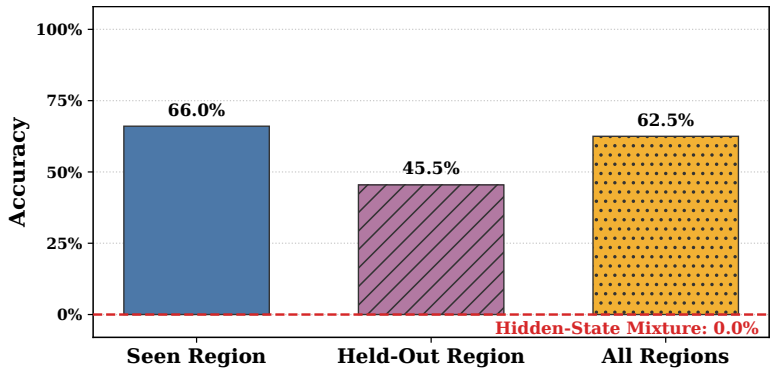


Figure 76: Qwen3-8B retains partial RGB accuracy in a contiguous region omitted from training, with a gap from seen-region performance.

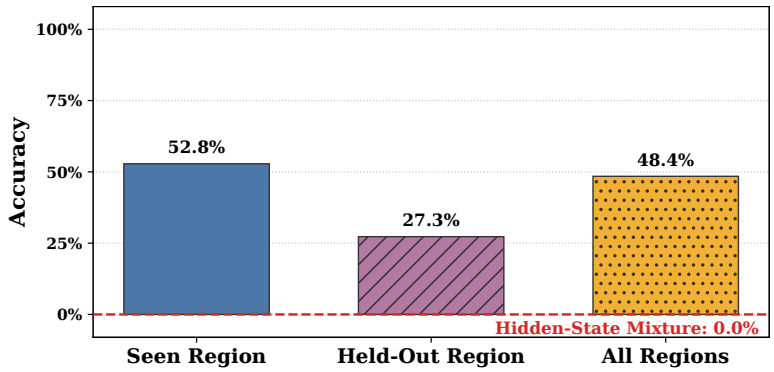


Figure 77: Qwen3-14B retains partial RGB accuracy in a contiguous region omitted from training, with a gap from seen-region performance.

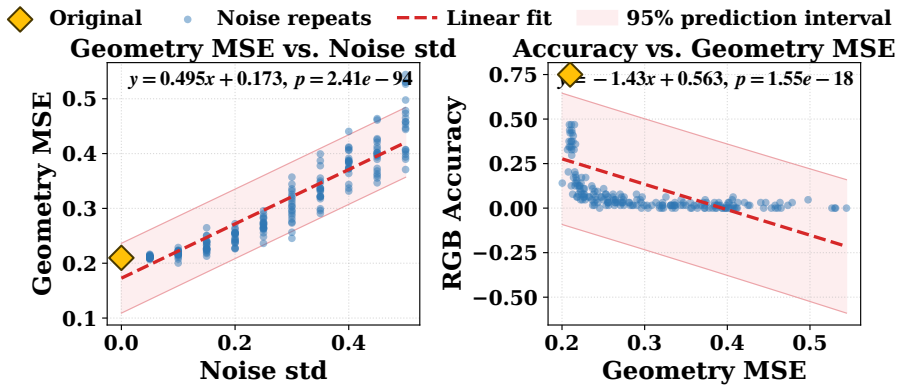


Figure 78: Perturbing the learned Llama3-8B coordinates degrades RGB output behavior as geometric organization is lost.

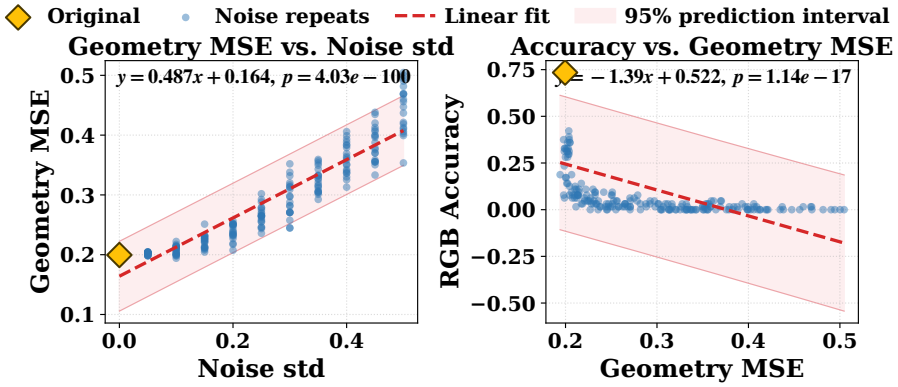


Figure 79: Perturbing the learned Llama3.1-8B coordinates degrades RGB output behavior as geometric organization is lost.

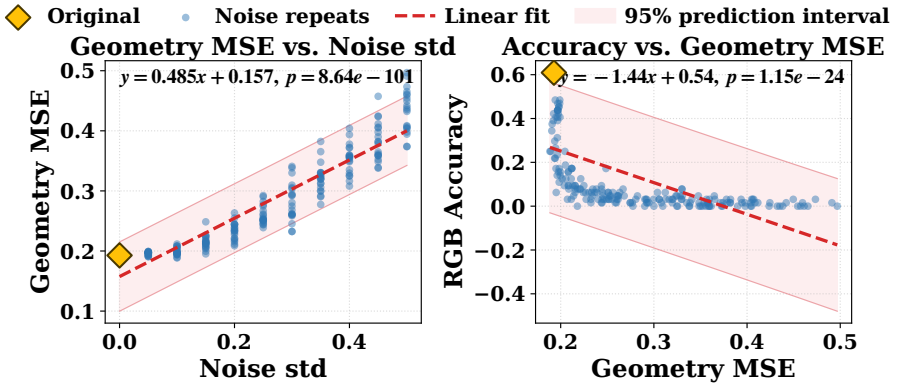


Figure 80: Perturbing the learned Llama3.2-3B coordinates degrades RGB output behavior as geometric organization is lost.

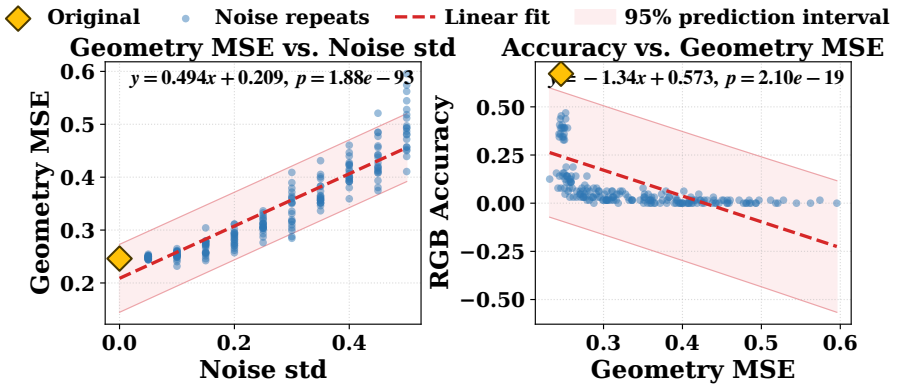


Figure 81: Perturbing the learned Llama3-70B coordinates degrades RGB output behavior as geometric organization is lost.

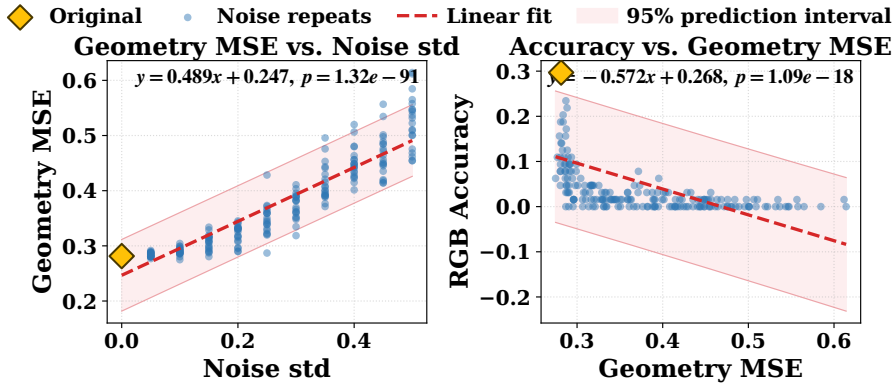


Figure 82: Perturbing the learned Qwen3-1.7B coordinates degrades RGB output behavior as geometric organization is lost.

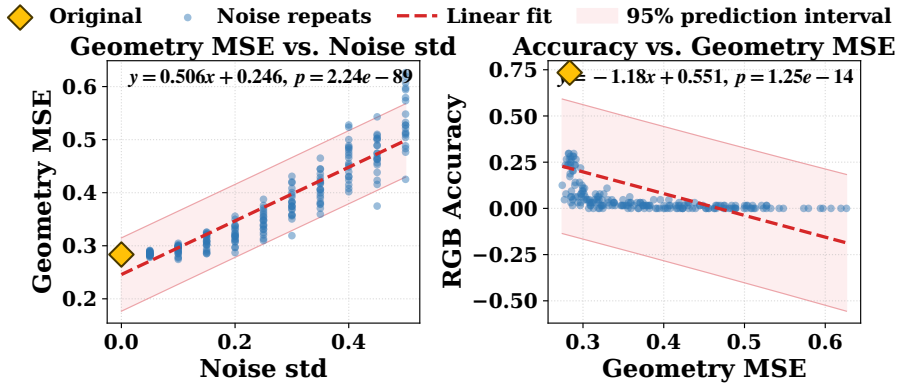


Figure 83: Perturbing the learned Qwen3-8B coordinates degrades RGB output behavior as geometric organization is lost.

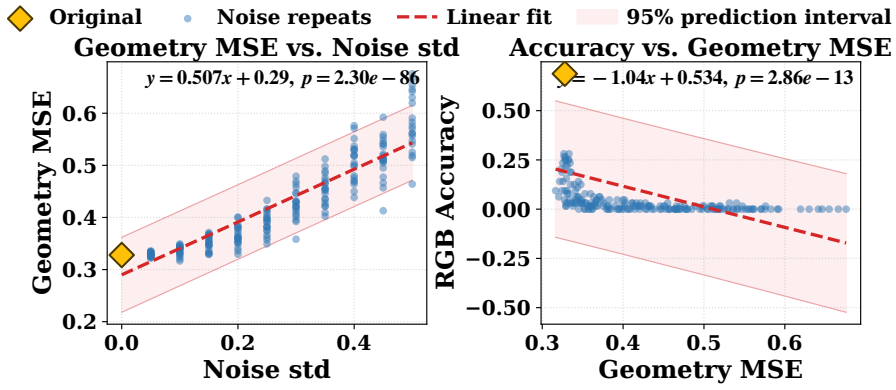


Figure 84: Perturbing the learned Qwen3-14B coordinates degrades RGB output behavior as geometric organization is lost.

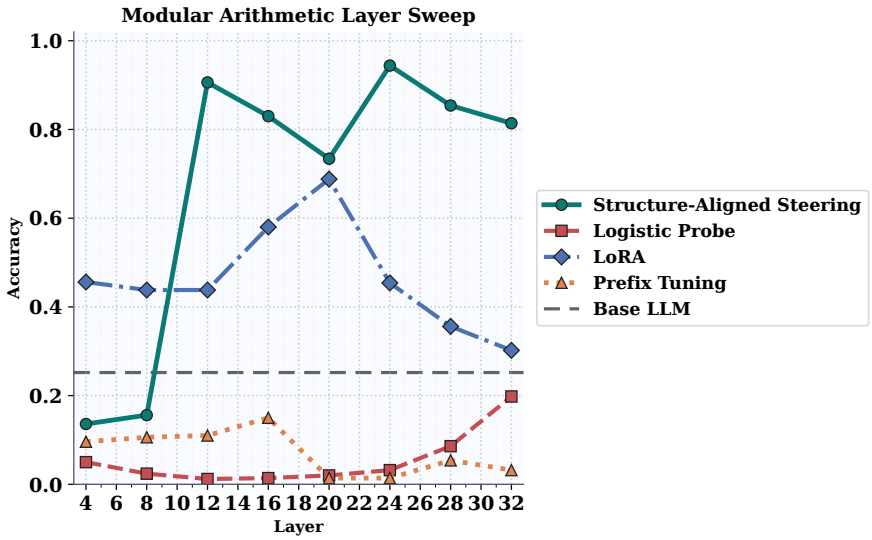


Figure 85: Qwen3-4B succeeds across middle and late layers but fails at layers 4 and 8, exposing a clear dependence on insertion depth.

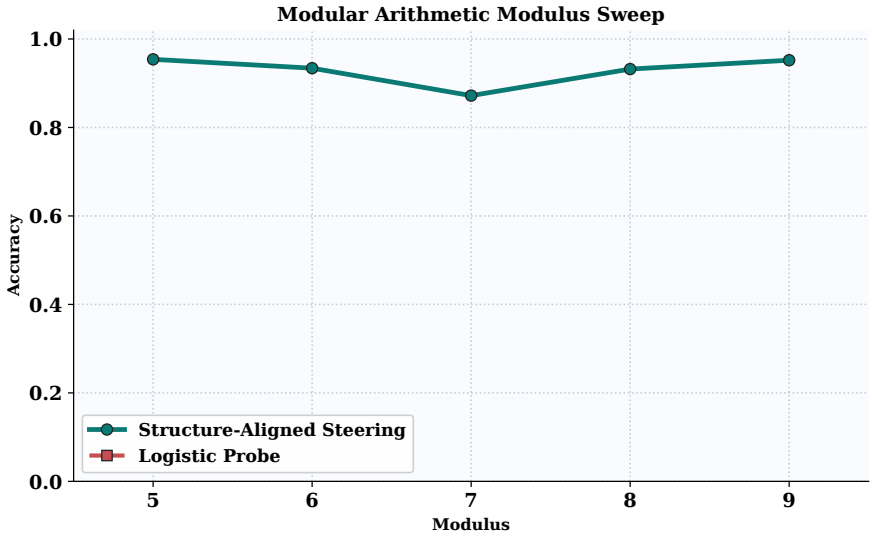


Figure 86: Retraining Qwen3-4B for moduli 5, 6, 8, and 9 preserves strong accuracy across different cyclic-group sizes.

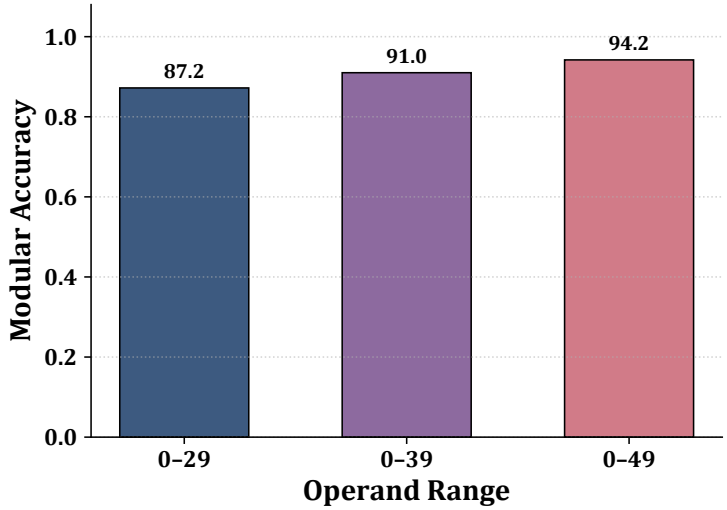


Figure 87: Retraining for larger operand vocabularies preserves strong modular-arithmetic accuracy.

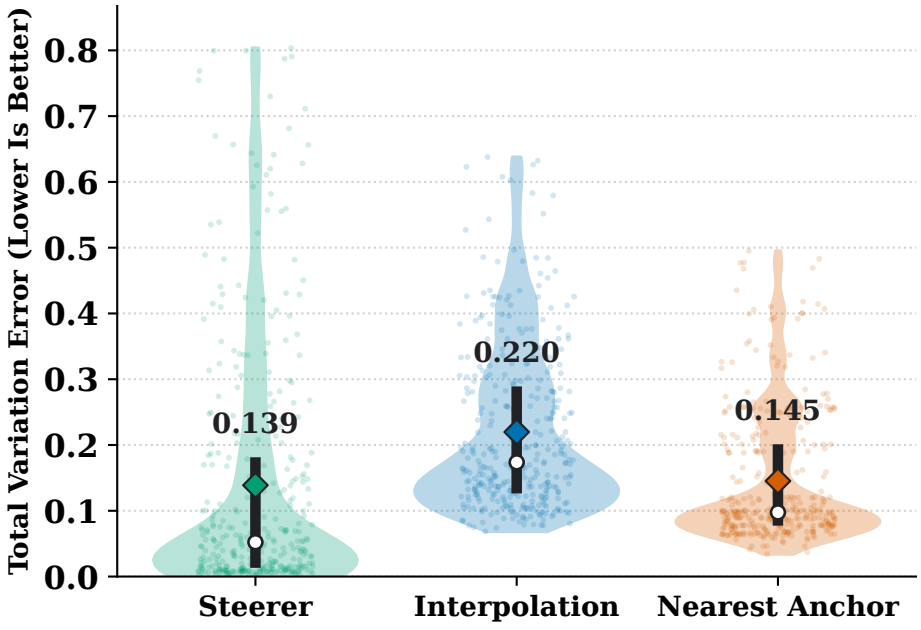


Figure 88: On Qwen3-4B, the Modular steerer yields lower phasewise TV error than direct interpolation or nearest-anchor snapping.

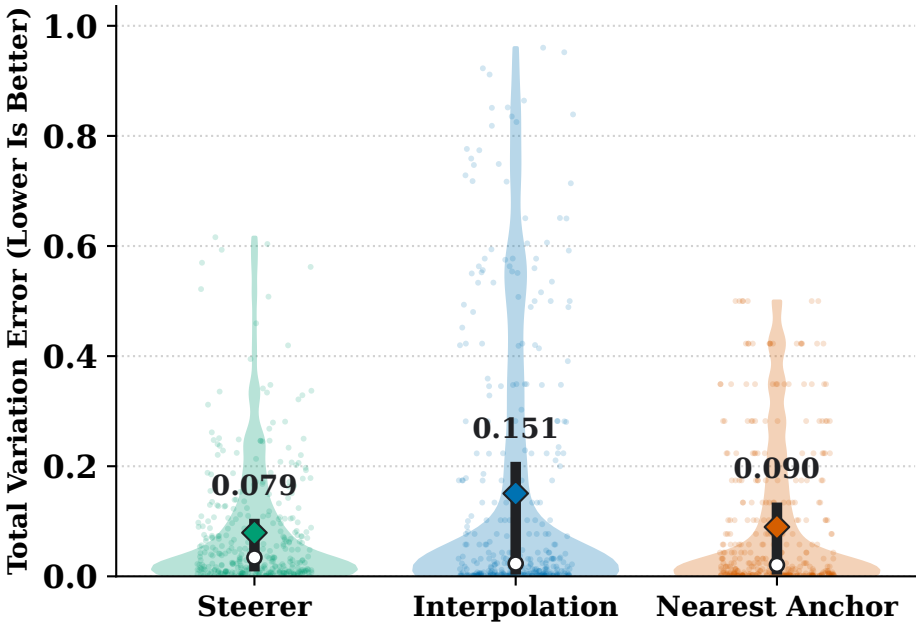


Figure 89: On Llama3-8B, the Modular steerer lowers phasewise TV error relative to direct interpolation and nearly matches nearest-anchor snapping.

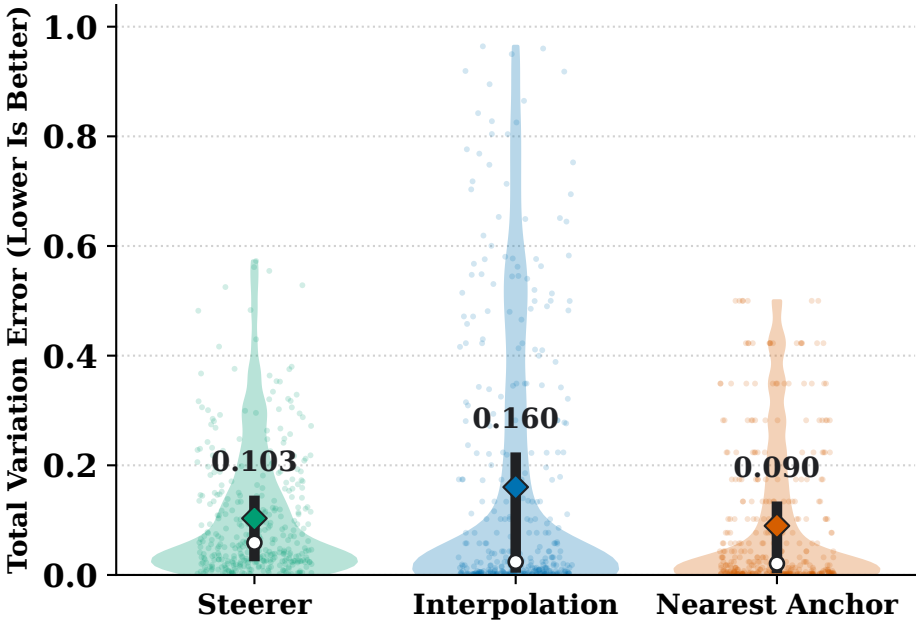


Figure 90: On Llama3.1-8B, the Modular steerer improves over direct interpolation, while nearest-anchor snapping has lower phasewise TV error.

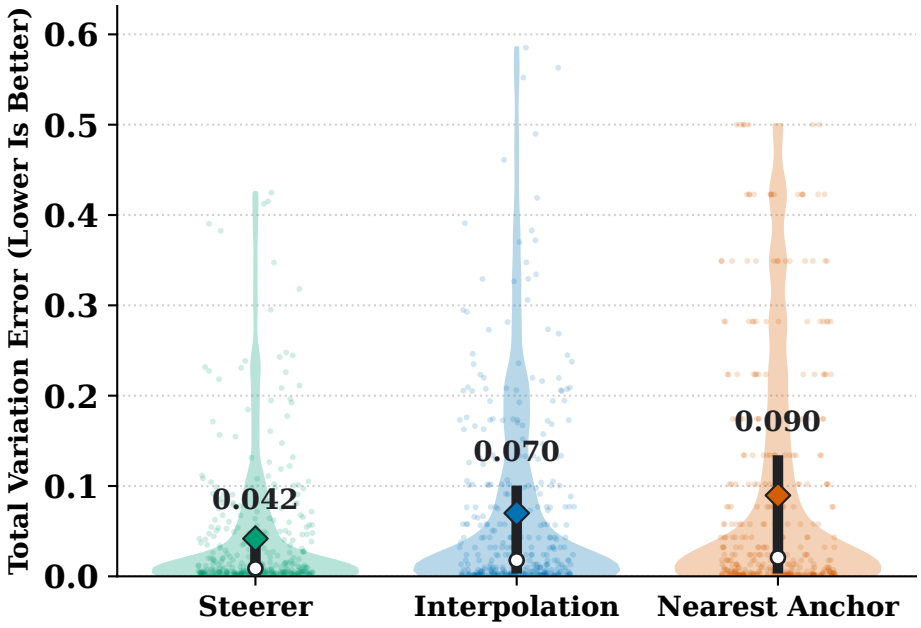


Figure 91: On Llama3.2-3B, the Modular steerer has the lowest phasewise TV error among the three readouts.

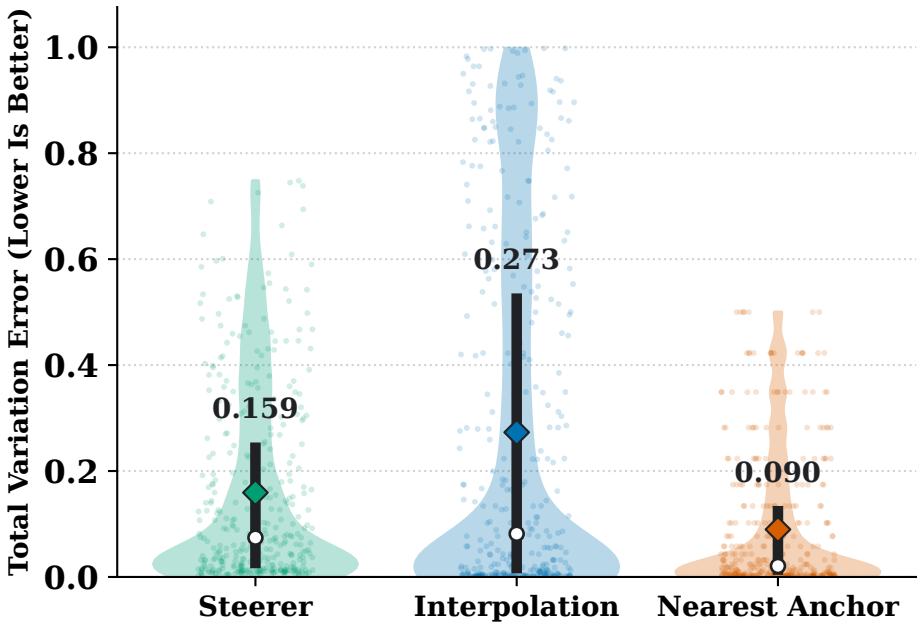


Figure 92: On Llama3-70B, the Modular steerer improves over direct interpolation, while nearest-anchor snapping has lower phasewise TV error.

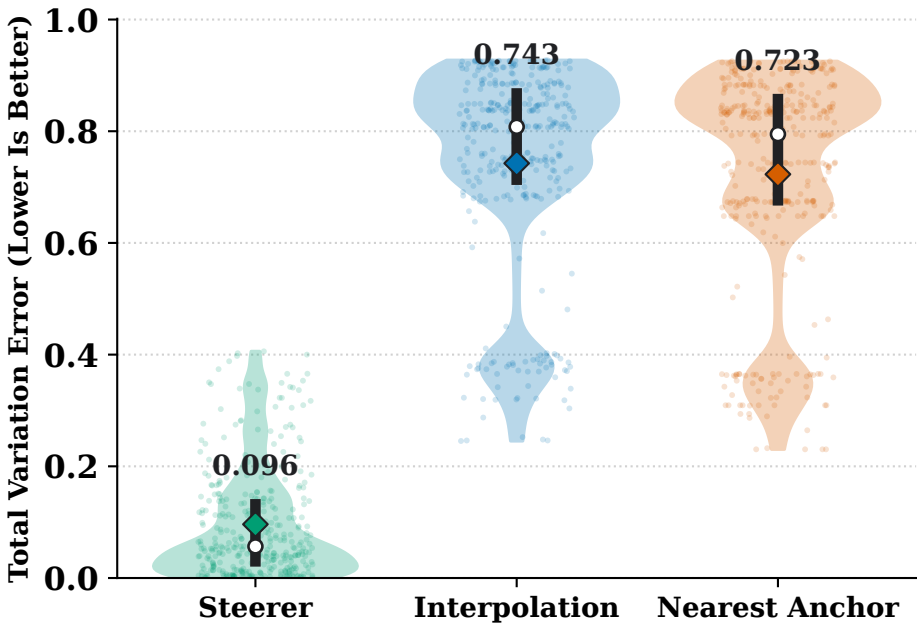


Figure 93: Across continuous residue phases, the Qwen3-1.7B Modular steerer induces lower target-distribution error than direct interpolation or nearest-anchor snapping.

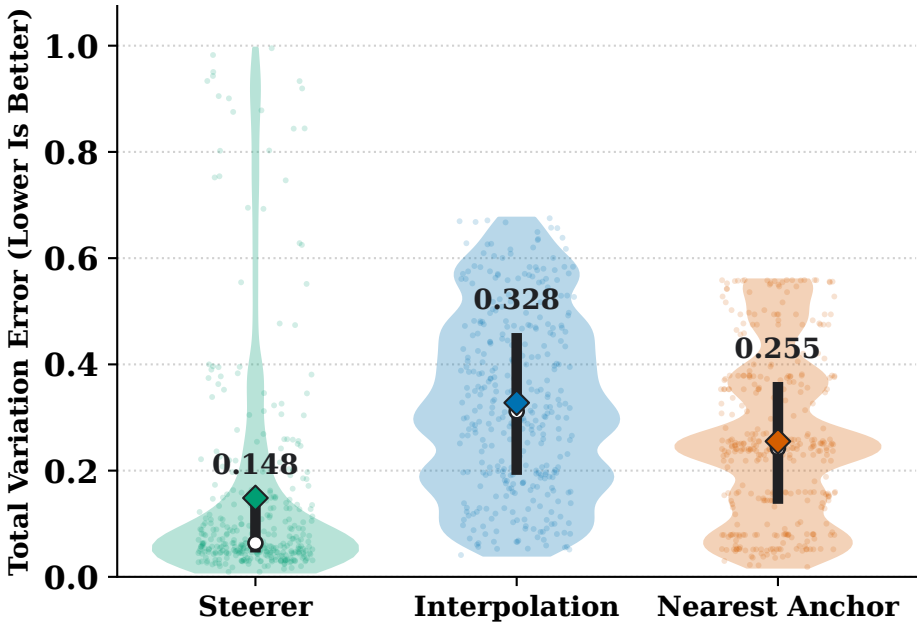


Figure 94: Across continuous residue phases, the Qwen3-8B Modular steerer induces lower target-distribution error than direct interpolation or nearest-anchor snapping.

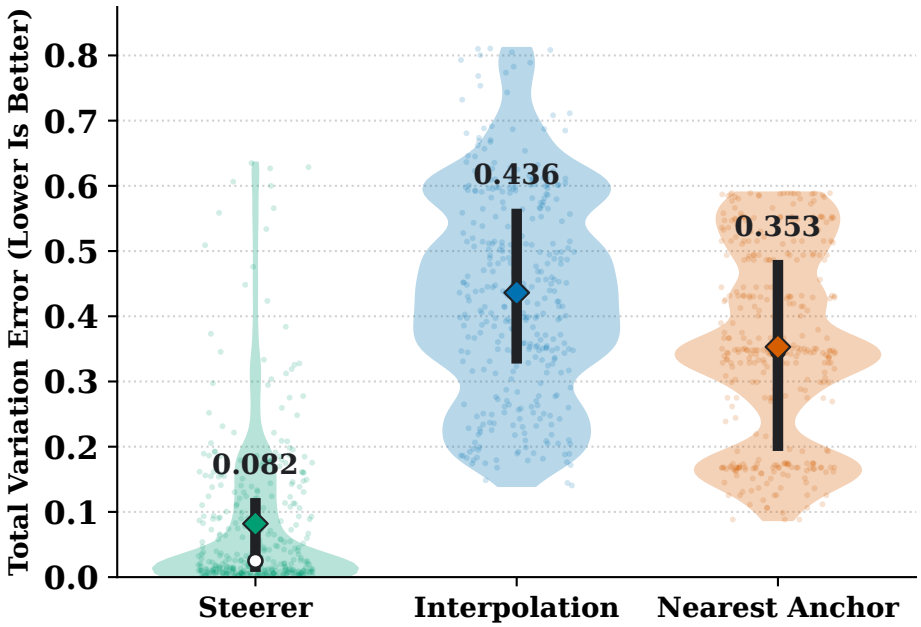


Figure 95: Across continuous residue phases, the Qwen3-14B Modular steerer induces lower target-distribution error than direct interpolation or nearest-anchor snapping.

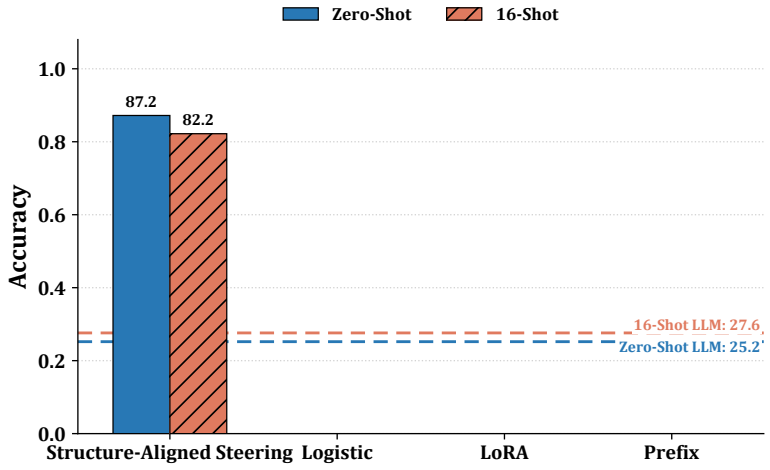


Figure 96: On Qwen3-4B, the separately trained 16-shot Structure-Aligned system reaches 82.2%, exceeding LoRA (51.0%), prefix tuning (10.8%), logistic regression (6.8%), and the frozen LLM (27.6%); zero-shot results provide the matched reference.

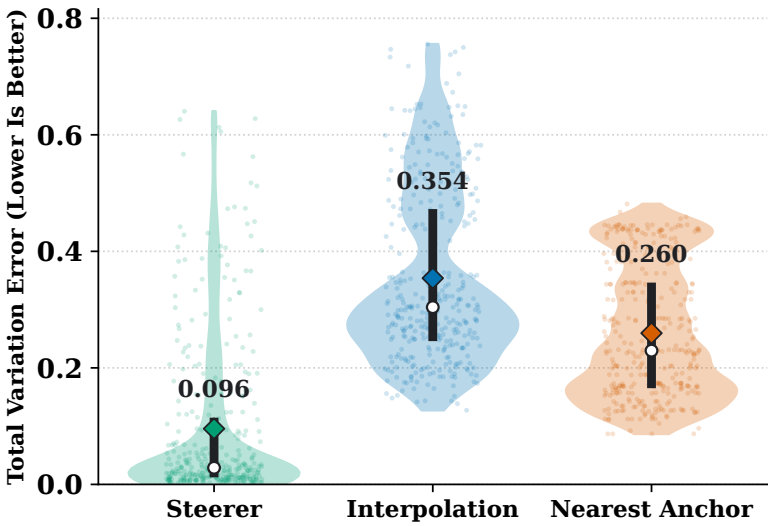


Figure 97: Repeating Figure 88 under Qwen3-4B 16-shot prompts, the steerer yields lower phasewise TV error (mean 0.096) than direct interpolation (0.354) or nearest-anchor snapping (0.260), preserving continuous circular output behavior with demonstrations.

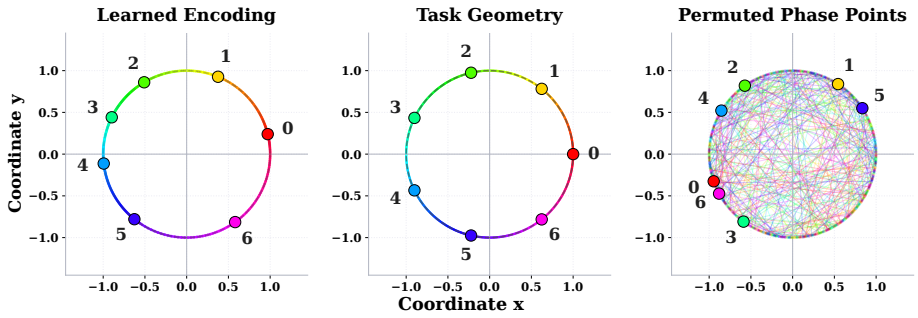


Figure 98: The Qwen3-4B Modular encoder organizes all 350 interpolated phases into an ordered closed trajectory (MSE 0.043), unlike a distribution-matched phase permutation (0.995 ± 0.037).

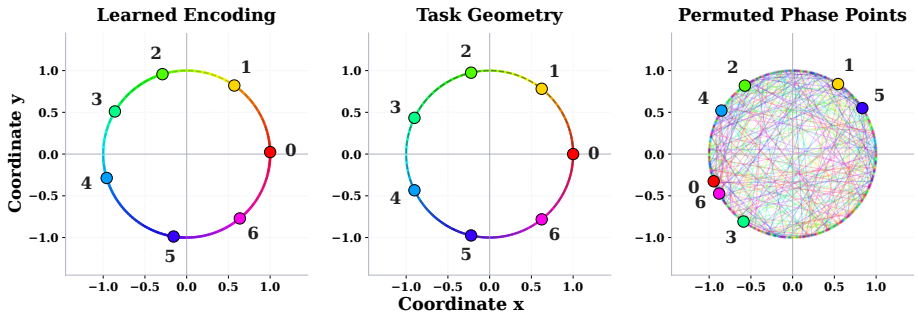


Figure 99: Llama3-8B closely recovers the complete Modular circle between residue anchors (MSE 0.005), while permutation destroys phase neighborhoods (0.995 ± 0.037).

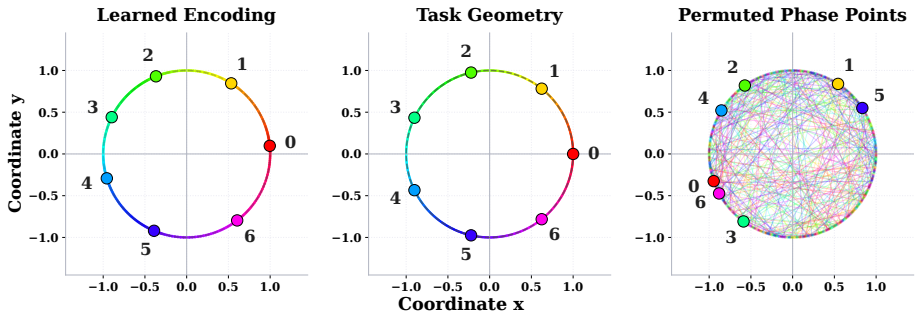


Figure 100: Llama3.1-8B preserves the cyclic order of interpolated Modular mixtures (MSE 0.010), far below the matched phase-permutation null (0.995 ± 0.037).

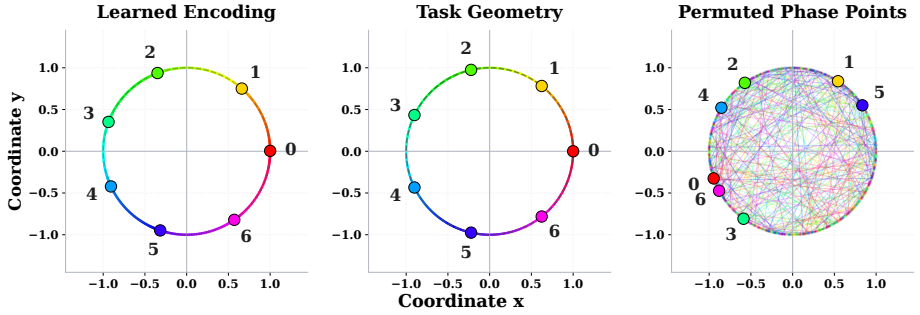


Figure 101: Llama3.2-3B reconstructs a continuous closed Modular trajectory (MSE 0.005), unlike the distribution-matched permutation control (0.995 ± 0.037).

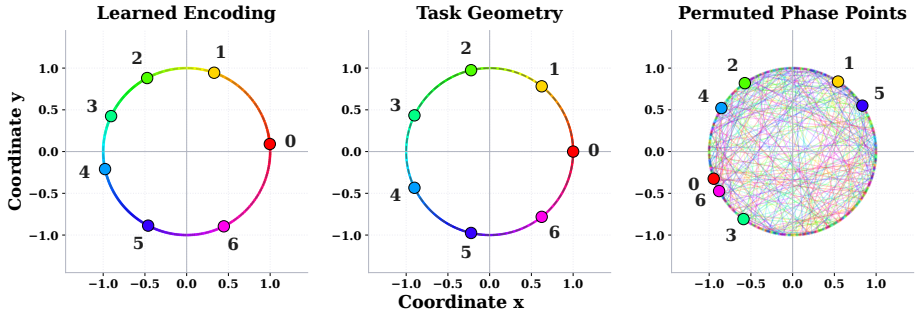


Figure 102: Llama3-70B organizes intermediate Modular phases around the reference circle (MSE 0.027), whereas permutation removes their ordered correspondence (0.995 ± 0.037).

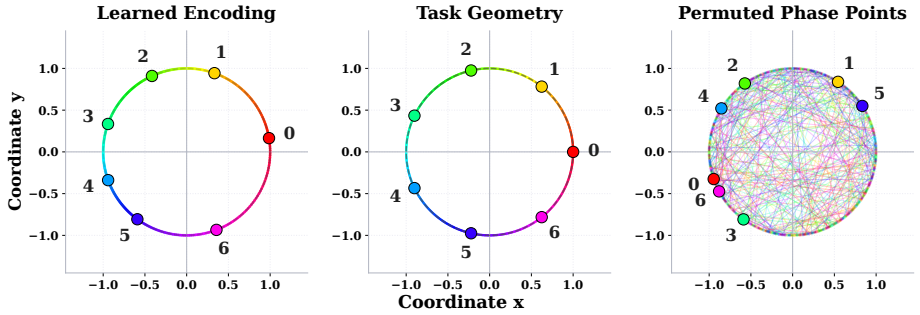


Figure 103: Qwen3-1.7B learns a closed, phase-ordered Modular realization (MSE 0.031), substantially closer to the task circle than permuted phase points (0.995 ± 0.037).

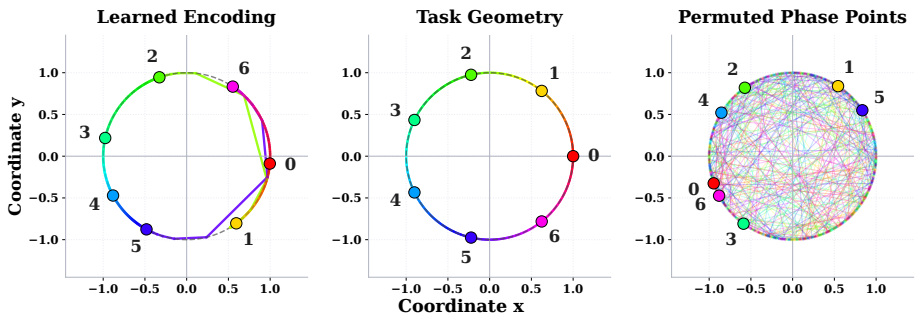


Figure 104: Qwen3-8B largely follows the Modular circle but exchanges residues 1 and 6, so a few interpolated arcs jump across the circle (MSE 0.470, versus 0.995 ± 0.037 for the permutation null).

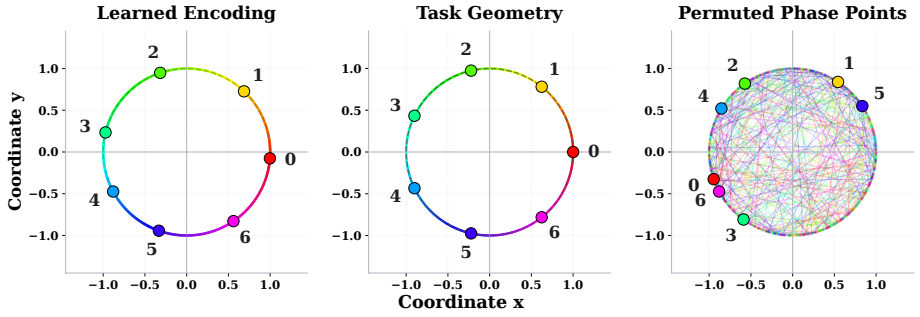


Figure 105: Qwen3-14B recovers the Modular circle with its residues in cyclic order (MSE 0.007, versus 0.995 ± 0.037 for the permutation null).

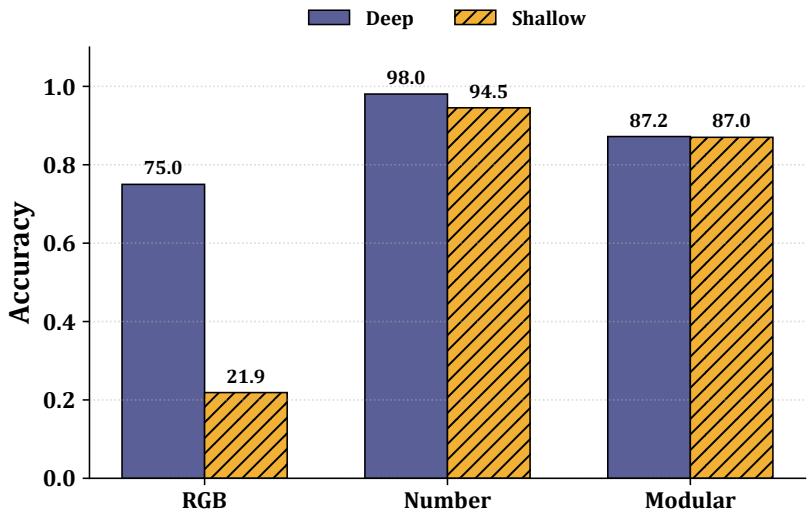


Figure 106: Additional nonlinear depth matters substantially more for the three-dimensional RGB interface than for the circular number and modular interfaces.