RIEMANNIAN MANIFOLD STEERING STAYS ON A DATA-INFORMED TRACK

Anonymous authors
Paper under double-blind review

ABSTRACT

Activation steering aims to control a model's behavior by moving activations along a linear direction or interpolating between concepts. However, steered activations can lie far from those the model naturally produces, degrading its output. Prior manifold-aware steering (Wurgaft et al., 2026) addresses this with cubic splines, but requires labeled concept centroids and assumptions about the manifold's topology. Instead, we model the activation space as a Riemannian manifold whose metric is pulled back from the output distributions. We consider geodesics (shortest paths) under that metric with a regularizing term that penalizes paths moving away from observed activations. We call this unsupervised method Riemannian manifold steering (RMS). We first examine path recovery and surface drift on synthetic surfaces with known geometry, isolating the roles of the metric and data support regularizer. We then show that the resulting paths produce outputs closer to the natural output distributions of the model across a range of settings, from toy examples to real-world LLM tasks.

1 INTRODUCTION

Activation steering (Rimsky et al., 2024; Dunefsky & Cohan, 2025; Oozeer et al., 2025) changes a model's behavior by intervening on its internal representations, either by moving activations along a steering direction or by interpolating between concepts. In both cases, we want the steered activations, including the intermediate points of an interpolation, to remain close to those the model naturally produces on data (Wurgaft et al., 2026). However, moving along a straight line in activation space can leave the region of natural representations, producing outputs that differ from the model's natural behavior. This can happen when representations follow curved structures (Raval et al., 2026; Gurnee et al., 2025). Figure 1 illustrates this on a torus, where a straight path between two points on the surface cuts through the hole and leaves the region where representations are observed.

Manifold steering addresses this problem by accounting for the structure of the representations. For example, Wurgaft et al. (2026) fit cubic splines through concept centroids, with different constructions depending on the topology, as in a closed loop for cyclic/periodic concepts and an open loop for sequential concepts. These paths can follow the representations more closely than linear interpolation, but require labeled centroids and a choice of manifold structure. More generally, linear steering restricts movement to straight paths, while other constructions impose different geometric assumptions, such as paths on a sphere (You et al., 2026). A model may represent different concepts through distinct geometries or hierarchical structures (Bunin et al., 2026; Raval et al., 2026; Park et al., 2025), so a construction suited to one setting does not generalize to an arbitrary setting. In addition, a model can also represent a combination of different manifolds with their distinct geometries within the same space (Bhalla et al., 2026; Shafran et al., 2026). This motivates steering methods that use the observed data and the model's map to outputs without requiring the manifold's structure to be specified in advance. We discuss other related work on activation steering, manifold learning, and Riemannian geometry in Appendix A.

Inspired by manifold-learning approaches such as GAGA (Sun et al., 2025), we formulate steering as path optimization under a Riemannian metric that measures changes in behavior rather than distances in activation space, since the two need not agree. On the torus in Figure 1, the map from activations to outputs stretches some regions of the surface (orange) and compresses others (blue), so the shortest steering path must be measured by the resulting change in outputs. We therefore

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.

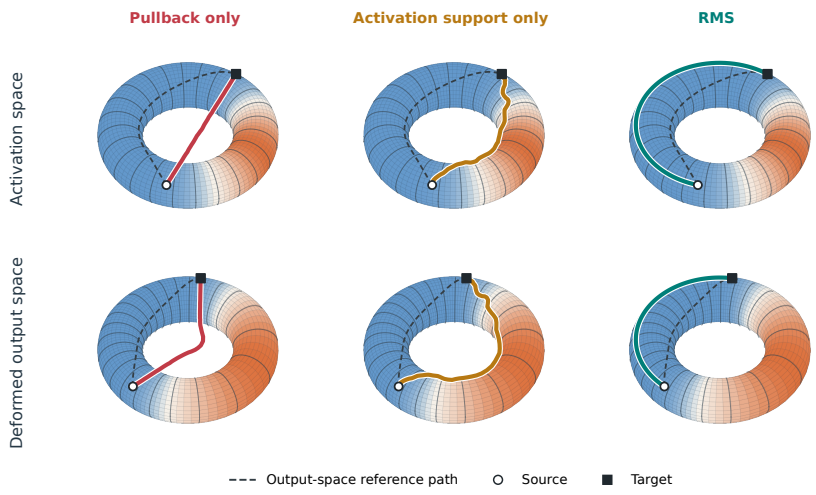

Figure 1: The top row shows paths in activation space; the bottom row shows them after deformation. Orange surface regions stretch, blue regions compress, and white regions stay unchanged. Solid lines show each method's path. Dashed lines show the output-space reference path, mapped back in the top row. Circles mark the start and squares mark the end. The deformation stretches one side of the torus, making the path around the other side shorter in output space. Pullback alone takes a shortcut that leaves the surface. Activation support alone keeps the path near the surface but is blind to this stretching. Together, they keep the path near the surface while avoiding the stretched region, as shown by RMS.

derive the metric from how the network transforms activations into outputs: the *pullback metric* carries the output-space metric into activation space, so that the length of an activation path equals the length of the output trajectory it induces (Amari, 2016; Benfenati et al., 2025). However, geodesics under this metric need not stay near the distribution of natural representations: in the first column of Figure 1, the pullback geodesic takes a shortcut that leaves the surface. To address this, we add an *activation-support penalty*, which discourages movement away from the observed activations. On its own, this term keeps the path near the surface but ignores the stretching (Figure 1, second column). Combining the two terms gives an objective that balances short paths under the pullback metric with proximity to observed activations. We call this approach *Riemannian manifold steering* (RMS) (Figure 1, third column).

This formulation is not tied to a particular geometry: the pullback metric and the support penalty can each be replaced by another Riemannian metric or distance from the data support, to suit the dataset at hand. Given a starting point and an endpoint, where the latter can be fixed in advance or left free, it constructs steering paths from activation samples and the model's map alone, without requiring intermediate centroid targets or a prescribed manifold parameterization.

Without a known manifold or geometry, however, steering paths are also difficult to evaluate. We therefore gather datasets with different levels of available ground truth. The synthetic surfaces let us compare our paths against known reference paths and measure drift from a known manifold. The weekday and age tasks provide known intermediate concepts, allowing us to examine behavior along paths without supplying these concepts as intermediate targets. Finally, we steer with a free endpoint on hallucination and sycophancy, a real-world setting where contrastive examples are available but the underlying manifold and intermediate concepts are unknown.

Our contribution is twofold:
1. Formulation and method. We define a steering objective that combines the pullback of an output-space metric to activation space with an activation-support penalty, and compute minimizing paths that require neither intermediate centroid supervision nor a prescribed manifold topology. Under idealized assumptions, we also explain how controlling deviations in activation space can bound deviations in output space.
2. Evaluation and findings. We evaluate against baselines on three settings with decreasing

amounts of known structure. We construct a synthetic dataset to evaluate how both parts of our formulation — pullback metric and activation-support penalty — affect steering. On controlled toy LLM tasks, we show that our method reduces mean distance to observed output distributions relative to spline steering. In an age-task example where the labelled centroids are located in denser parts of the data distributions, our path are closer to the labelled centroids than a spline refitted after removing an intermediate centroid.

2 DATASET DESCRIPTIONS

Following the three levels of ground truth outlined in Section 1, we use one group of datasets per level, each with its own evaluation measures. On synthetic surfaces (Section 2.1), we measure how closely a steering path follows the known reference path and how far it moves from the surface. On the weekday and age tasks (Section 2.2), we compare behavior along paths against the known intermediate concepts and their order. On the real world dataset task (Section 2.3), where neither the manifold nor the intermediate path is known, we measure whether steering moves the model toward the target answer and how close steered outputs stay to outputs the model naturally produces. Each subsection follows the same structure: we first list the criteria that guided our choice of datasets, then describe the datasets and how we construct them.

2.1 SYNTHETIC DATASETS

We construct the synthetic tasks around three criteria:

1. **Reference paths.** As reference paths, we use shortest paths on the known surfaces and on deformed versions of them. These are available in closed form for some surfaces, such as the sphere, and approximated numerically for the rest. We map the output-space reference paths back to activation space to compare them with our steering paths.
2. **Known surfaces.** The activation surface is known, so we can measure how far a steering path leaves it. We include different curvatures and topologies to test paths on surfaces with and without holes.
3. **Separating two ways paths can go wrong.** The tasks separate paths that stay on the activation surface but are longer in output space from paths that take a short output-space shortcut by leaving the activation surface. A successful path must account for both output-space distance and data support.

We use four surfaces: a torus, a sphere, a hyperboloid, and a genus-2 surface (Appendix Figure 8). We embed samples from these three-dimensional surfaces in a higher-dimensional activation space and add noise. A known deformation, defined in Appendix D.1.1, plays the role of the model's map to outputs. It changes each point's angle around the vertical axis, spreading points farther apart around the axis in one region and bringing them closer together elsewhere, while keeping their height and distance from the axis unchanged (Appendix Figure 9). Consequently, a short path in activation space can become longer than an alternative path in output space. This lets us test whether steering accounts for how the model changes distances, rather than only following the activation surface.

Each surface stresses this test in a different way. On the torus, the deformation determines which way around the hole the preferred path goes. The sphere tests whether the deformation alone changes the preferred path; it also admits an exact output-space reference path. The hyperboloid introduces negative curvature and a narrow waist, where a straight shortcut can leave the surface. The genus-2 surface has two holes, so the path must choose among several routes around them. Dataset construction and reference path details are given in Appendix D.1; the evaluation measures are defined in Section 4.

2.2 TOY LLM DATASETS

We choose the toy LLM tasks around two criteria:

1. **Known concept structure.** We study one-dimensional concepts with either a cyclic order, as in days of the week, or a sequential order, as in ages.

2. **Known intermediate concepts.** The ordering tells us which concepts lie between the endpoints. We can therefore examine whether a steering path passes near their representations, produces probability peaks at intermediate concepts, and stays close to the observed output distributions.

We use weekday and age arithmetic tasks. Prior work identifies circular weekday representations (Engels et al., 2025) and models weekday and age representations with periodic and open curves, respectively (Wurgaft et al., 2026). Both tasks come with known intermediate concepts: for example, moving forward from Monday to Thursday passes through Tuesday and Wednesday. These let us examine how outputs change along a path, beyond whether it reaches the final answer.

To construct the tasks, we collect activations and output distributions from a set of prompt templates. We average activations from prompts with the same answer at a fixed layer to obtain one activation centroid per weekday or age, and decode these centroids through the model, representing their outputs in Hellinger coordinates. We fix two activation centroids as endpoints and optimize the path between them. Example templates and the use of the training and test splits are given in Appendices C.1.1 and C.1.2.

2.3 Real-World Datasets

We choose the real-world datasets according to two criteria:

1. **Contrasting examples of a behavior.** Each question has two answers, one exhibiting the behavior and one not. These pairs give the labels against which we measure whether steering moves the model toward the desired answer.
2. **Unknown geometry and intermediate concepts.** These datasets do not provide a manifold, its topology, or an ordered set of intermediate concepts. They let us study steering when we know the desired behavioral change but not the path it should follow.

The datasets of Rimsky et al. (2024) provide this setting. The difference between the activation means of the two groups of answers defines the Contrastive Activation Addition (CAA) steering direction. Unlike the centroids in the toy tasks, these means only summarize the labeled training examples: they do not fix the endpoints of our steering paths. For the hallucination experiment, we use Llama-2-7b-chat (Touvron et al., 2023) with open-ended questions and their paired answer texts. Each path starts from the prompt's natural activation, with its endpoint initialized along the CAA direction and then optimized. Besides reaching the target answer, we check that steering does not push the model's outputs away from the observed output distribution. Prompt selection, activation extraction, and the training examples used for the two means are described in Appendix E.1.

3 Riemannian Manifold Steering

In this section, we formulate steering as finding a short path under a chosen Riemannian metric. Steering methods differ in how they move between two activations, but each implicitly makes two choices: a metric, which sets the cost of moving in each direction, and an admissible region, which sets where the path may go. In Section 3.1, we make these choices explicit and show that linear steering (Euclidean metric, whole activation space) and parametrized steering (both taken from a prescribed parameterization of the manifold) fit within this framework. This also shows what must be replaced when the manifold's geometry is not known in advance. We replace each choice with one grounded in the model and its data. In Section 3.2, we introduce a pullback metric from the output space, so that path length measures behavioral change. In Section 3.3, we add a support penalty estimated from observed activations, which encourages paths to remain near the data, and describe how we compute the resulting paths. Together, these define our method, ***RMS***.

Notation. We write $\mathbb{R}^n$ for the ambient activation space and $\mathcal{M} \subset \mathbb{R}^n$ for the d-dimensional data manifold. Observed activations x_j^{obs}, $j = 1, \ldots, m$, are the model's activations on a set of m data samples. We use them to estimate the support of the activation data. A steering path $\gamma : [0, 1] \to \mathbb{R}^n$ connects endpoints x_0, x_1, which are class centroids in our toy LLM experiments; Γ_{x_0,x_1} denotes the set of such Lipschitz paths with fixed endpoints. The map F sends activations to the chosen

output representation, and $g(x)$ is a symmetric positive-definite matrix representing the Riemannian metric at $x \in \mathbb{R}^n$. We use $L[\gamma]$ for path length. The support penalty $r(x)$ measures how far an activation x lies from the observed data support. We write $R[\gamma] = \int_0^1 r(\gamma(t))\,dt$ for the average penalty over the path parameter, weighted by $\lambda > 0$.

3.1 Steering under a chosen geometry

A Riemannian metric $g(x)$ on the ambient activation space assigns a cost to each direction of movement at x. For a path $\gamma \in \Gamma_{x_0,x_1}$, its length is

$$L[\gamma] = \int_0^1 \sqrt{\dot\gamma(t)^\top g(\gamma(t))\dot\gamma(t)}\,dt. \tag{1}$$

Ideally, to find the shortest path connecting x_0 with x_1, we minimize $L[\gamma]$ subject to $\gamma(t) \in \mathcal{M}$ (do Carmo, 2016). Appendix B.1 formalizes this constraint in Equation 9. The metric determines movement cost; when working in an ambient space the admissible space determines where the path may go.

Linear steering allows movement throughout $\mathbb{R}^n$ with $g = I$. Its shortest path is $\gamma(t) = (1-t)x_0 + tx_1$, which can leave a curved data manifold. *Parametrized* steering instead specifies an admissible region through a smooth embedding $\chi : U \to \mathcal{M}$, with $U \subseteq \mathbb{R}^d$. Choosing a metric with $(D\chi_u a)^\top g(\chi(u))(D\chi_u b) = a^\top b$ makes parameter coordinates Euclidean. Then $\chi((1-t)u_0 + tu_1)$ is shortest within that region when the parameter segment lies in U, with $x_0 = \chi(u_0)$ and $x_1 = \chi(u_1)$. Periodic coordinates also require a choice of branch. This perspective (Wurgaft et al., 2026) exposes the geometry supplied by the parameterization. We instead derive movement cost from the model's map.

3.2 Measuring output change with a pullback metric

Let $F : \mathbb{R}^n \to \mathbb{R}^k$ be a frozen differentiable map into coordinates where Euclidean distance measures output change. We use

$$g(x) = J_F(x)^\top J_F(x) + \epsilon I, \qquad v^\top g(x) v = \|J_F(x)v\|_2^2 + \epsilon\|v\|_2^2, \quad \epsilon > 0. \tag{2}$$

The pullback term $J_F(x)^\top J_F(x)$ measures how a small activation displacement changes the output (Benfenati & Marta, 2023a;b; Benfenati et al., 2025). The positive floor charges directions invisible to F, if any, and makes g positive definite—also absorbing, for ϵ above the rounding level, the small negative eigenvalues that floating-point errors can introduce. This ties the geometry to the model's response, without prescribing a parameterization of $\mathcal{M}$. Using this metric, $L[\gamma]$ measures movement along the path through its output changes, together with the small activation-space contribution.

For LLMs, $p_c(x)$ is the output distribution after inserting x into a fixed prompt context c. We choose

$$F(x) = F_c(x) = \frac{\sqrt{p_c(x)}}{\sqrt{2}}, \qquad \|F_c(x) - F_c(y)\|_2 = H(p_c(x), p_c(y)), \tag{3}$$

where $H(p,q) = \|\sqrt{p} - \sqrt{q}\|_2/\sqrt{2}$ is Hellinger distance. For synthetic tasks, we use $F = f_\theta$, a learned encoder approximating distances under a known deformation Φ (Appendix D.2.1).

3.3 Keeping paths near the observed activations

Low output-change cost alone does not ensure that a path remains near observed activations. Appendix B.1 relaxes the requirement that paths remain on $\mathcal{M}$ using a finite support penalty and a constant-speed constraint:

$$\begin{aligned} &\min_{\gamma\in\Gamma_{x_0,x_1}} L[\gamma] + \lambda R[\gamma], \qquad R[\gamma] = \int_0^1 r(\gamma(t))\,dt, \\ &\text{subject to } \sqrt{\dot\gamma(t)^\top g(\gamma(t))\dot\gamma(t)} = L[\gamma] \quad \text{for a.e. } t \in [0,1]. \end{aligned} \tag{4}$$

We analyze this idealized formulation with constant metric speed. In practice, we approximate the objective using uniformly spaced parameter samples, without explicitly enforcing constant speed.

The numerical approximations and checks of the estimated LLM path lengths are described in Appendices B.3 and C.2.2.

We estimate $r(x)$ from the observed activations x_j^{obs}, rather than reconstructing the entire manifold. For synthetic data, we combine nearest-neighbor distance within a low-dimensional projection with distance from that subspace (Appendix B.4.1). For LLMs, we estimate nearest-neighbor distances using a graph weighted by pullback lengths (Appendix B.4.2). These choices require neither intermediate centroid targets nor a prescribed topology. The pullback-only ablation sets $\lambda = 0$. Appendix B.2 shows how an idealized support penalty can bound the largest deviation from the manifold in activation and output space.

Computing the paths. We discretize γ into waypoints $\gamma_i \approx \gamma(i/N)$, with fixed endpoints $\gamma_0 = x_0$ and $\gamma_N = x_1$. Appendix B.3 gives the midpoint approximation (Equation 22):

$$\underset{\gamma_1,\ldots,\gamma_{N-1}}{\arg\min} \sum_{i=1}^{N} \sqrt{(\gamma_i - \gamma_{i-1})^\top g(\bar{\gamma}_i)(\gamma_i - \gamma_{i-1})} + \frac{\lambda}{N+1} \sum_{i=0}^{N} r(\gamma_i), \tag{5}$$

where $\bar{\gamma}_i = \frac{1}{2}(\gamma_{i-1} + \gamma_i)$. This discretization gives a finite-dimensional problem that can be solved using numerical optimization methods. For synthetic experiments, we use the energy variant in Appendix D.3.1 and optimize the intermediate waypoints using L-BFGS (Liu & Nocedal, 1989), with several initial paths informed by the known geometry (Appendix D.3.3). Such initial paths are harder to construct for LLM activations, where the manifold's structure is unknown. We therefore use a neural bridge to parameterize the path and train its parameters while keeping both endpoints fixed and the language model frozen (Appendix C.2.3). We approximate length using successive mapped outputs, as defined in Equation 32. The Steering in the Wild experiments in Section 5.3 use a modified objective and bridge with only the starting activation fixed.

4 Evaluating Steering Trajectories

We compare paths produced by RMS (Section 3) with the baselines on the datasets introduced in Section 2. We define the evaluation metrics for each setting below and use them to report the results in Section 5.

Synthetic experiments. We compare paths with the activation-space and output-space reference paths from Section 2.1, mapping the latter back to activation space. Endpoints are chosen so these reference paths are clearly different. **Path recovery** checks whether a path follows the output-space reference path's direction; for genus-2, it checks whether the path is closer to that reference path than to the activation-space one. **Path deviation** measures the average activation-space distance between points at the same fraction of the distance along our path and the output-space reference path. **Surface drift** is the largest estimated distance to the known, noise-free surface across sampled points, including between optimized waypoints. The first two measures test whether steering accounts for the deformation; the third checks whether paths stay near the surface. Formulas are given in Appendix D.5, Equations 51–53; reference path construction and endpoint selection are described in Appendices D.4.1 and D.4.2.

Toy LLM experiments. We use two proximity scores. **Activation pullback distance** averages distances to the nearest observed activations using the pullback graph. **Output Hellinger distance** averages Hellinger distances between a decoded output and its nearest observed output distributions, selecting neighbors independently of the activation graph. Lower scores mean closer proximity. The output-to-data distance is not part of our optimization objective. For toy tasks, outputs are grouped into concept bins plus a bin for other tokens, and both scores are averaged along the path (Appendix C.2.6).

Intermediate peaks. count non-endpoint weekdays whose probability reaches its maximum inside the path; a peak is *dominant* if that weekday also exceeds all other weekdays. These counts do not establish calendar order (Appendix C.4). **Omitting a centroid** tests reliance on intermediate guidance: we remove an age centroid and refit only the spline, keeping all observed data available. We compare the same proximity scores at the omitted centroid's original path position and over the whole path (Appendix C.5).

Real-world experiments We compare RMS with optimized straight paths under the same objective and tuned mean-difference steering (Appendix E.3). We use the output Hellinger score from Section 4, but compare **full-vocabulary distributions at the endpoint**, with observed activations decoded under the same prompt. **Steering strength** measures preference for the target answer over the contrasting answer through the difference in their mean token log probabilities. Higher values favor the target behavior; for hallucination, this means favoring hallucinated answers, not factual accuracy. Both endpoint scores are averaged over prompts. Formulas and evaluation details are given in Appendices E.4 and E.5.

5 RESULTS

5.1 SYNTHETIC EXPERIMENTS

We compare RMS with activation support only and pullback only on the four surfaces of Section 2.1, using the measures of Section 4. RMS meets the path-recovery criterion on all six pairs per surface, compared with none for activation support only (Table 1). Pullback only recovers all six paths on the sphere and genus-2, but none on the torus and hyperboloid. Figure 1 illustrates the two failure modes: activation support alone ignores the deformation, while pullback alone takes a shortcut off the surface.

RMS also lowers median surface drift relative to pullback only on all four surfaces, and median path deviation relative to both baselines on the torus, sphere, and hyperboloid. On genus-2, pullback only has lower deviation but larger surface drift. Activation support only has slightly lower drift on the sphere and hyperboloid, but recovers neither surface's reference paths. Together, the results show why accounting for output geometry and staying near the activation surface are complementary requirements. Endpoint selection and numerical details are given in Appendix D.4.

Table 1: Synthetic path recovery, median surface drift, and median path deviation. Higher recovery and lower distances are better (Section 4). Bold scores mark the best value within each surface, including ties.

Method	Path recovery ↑	Surface drift ↓	Path deviation ↓
Torus			
Activation support only	0/6	0.01687	0.82502
Pullback only	0/6	0.55700	0.55078
RMS	**6/6**	**0.00158**	**0.26613**
Sphere			
Activation support only	0/6	**0.00175**	1.26672
Pullback only	**6/6**	0.01850	0.00713
RMS	**6/6**	0.00178	**0.00142**
Hyperboloid			
Activation support only	0/6	**0.00081**	0.63336
Pullback only	0/6	0.41316	0.38283
RMS	**6/6**	0.00102	**0.00082**
Genus-2			
Activation support only	0/6	0.01439	0.66497
Pullback only	**6/6**	0.01941	**0.04413**
RMS	**6/6**	**0.00455**	0.05348

5.2 TOY LLM MANIFOLDS

In this section, we test our method on toy LLM tasks with known intermediate concepts (Section 2.2), where we want steering paths to stay close to these concepts in output space. Spline steering requires the intermediate centroids, their order, and a choice of periodic or open traversal; our method uses only the two endpoint centroids. We compare both with linear steering (Section 3.1) using the measures of Section 4; further details are in Appendices C.2 and C.7.

Table 2: Distance to observed outputs and activations (mean $\pm$ SE across 21 paths per method per task). Lower is closer. RMS has the lowest distance on both tasks.

Method	Output Hellinger ↓	Activation pullback ↓
Weekdays		
Linear	0.09728 ± 0.00110	0.10850 ± 0.00108
Spline steering	0.09646 ± 0.00107	0.10821 ± 0.00126
RMS	$\mathbf{0.08244} \pm 0.00187$	$\mathbf{0.08364} \pm 0.00164$
Ages		
Linear	0.27927 ± 0.00467	0.26394 ± 0.00378
Spline steering	0.26698 ± 0.00146	0.24496 ± 0.00159
RMS	$\mathbf{0.23843} \pm 0.00445$	$\mathbf{0.21737} \pm 0.00487$

Our paths stay closer to the observed outputs. Across the weekday and age tasks, our paths stay closer to the observed outputs than linear and spline steering (Table 2) inducing probability

distributions of our decoded paths to peak at intermediate concepts on the known output manifold (Table 4). We measure proximity to the activation space and output space distributions using the Activation Pullback and the Output Hellinger Distances introduced in Section 4. We also show that when the centroids lie in denser regions such as the age manifold, we are able to stay closer to the intermediate concept centroids than both our baselines (spline steering and linear steering) as shown in Figure 2. To demonstrate this, we consider a case where one intermediate centroid is unavailable to the spline which is the only method that is constructed to go through centroids. We refit the spline and compare all methods at the omitted centroid's original position along the path.

Staying on the data manifold keeps outputs natural. These results show that staying on the data manifold in the activation space allows us to stay on the output manifold in the output space and our paths produce distributions that are closer to the natural distributions of our model, providing empirical evidence to our theoretical arguments that an idealized support penalty can bound the largest deviation from the manifold in both the activation and the output space.

Our paths do not always pass through centroids. We also note that RMS does not always prefer paths that go through centroids in the activation and output spaces. In particular, we observed that this happens when the centroids themselves are not located on densest regions of the data distribution. For example, in the weekday task, the centroids lie in fairly dense regions of the reference data, but not the densest (full-space KNN measure measurements and densMAP visualization in Appendix C.1.1). As a consequence, our paths reach points with lower KNN distances than the endpoint centroids on average, showing that they can pass through more densely sampled regions (Appendix Figure 4).

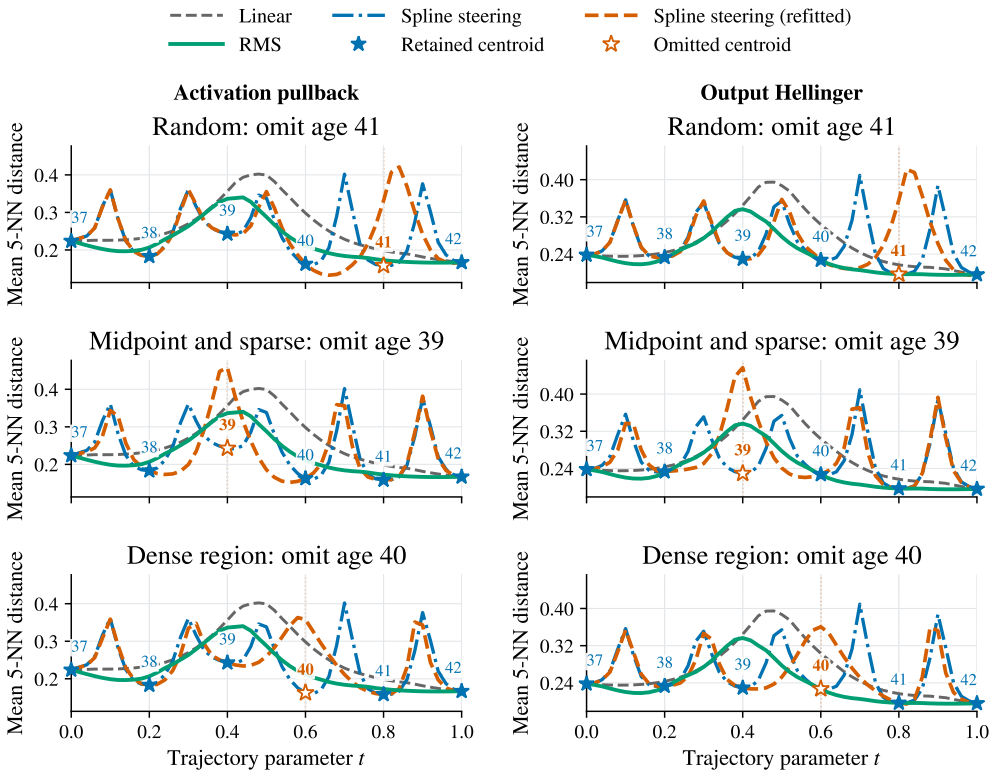

Figure 2: Leaving out an intermediate age centroid changes the spline path. The midpoint and sparse rules both select age 39 and share one row. The orange curve shows the refitted spline; the blue curve shows the original spline. Filled and hollow stars mark retained and omitted centroids, respectively. The left and right columns show distance to reference activations and outputs; lower is closer.

5.3 STEERING IN THE WILD

We now consider steering in a real world task where we have limited knowledge about the geometry. Since we don't know intermediate points that our paths should pass through, we therefore measure how our method performs with regards to increasing the steering strength (inducing the desired behavior) and how grounded are our decoded paths to the output distributions. We use the calibration prompts for Llama-2-7b-chat described in Section 2.3, using the baselines and evaluation metrics introduced in Section 4.

Allowing the endpoint to move. For a prompt q, we fix the natural starting activation x_q and optimize both the path and its endpoint. This allows us to reach an endpoint with a higher steering score. We use the length and support terms from Equation 4 without its constant-speed constraint, and add a reward for endpoint steering strength:

$$\min_{\gamma:\,\gamma(0)=x_q} L[\gamma] + \lambda R[\gamma] - \beta\big[b_q(\gamma(1)) - b_q(x_q)\big]. \tag{6}$$

Here b_q is the steering strength score and $\beta > 0$ sets the reward's weight. The objective balances the desired change in behavior against pullback length and the activation-support penalty. The neural bridge now has a trainable endpoint, while its start remains fixed; the change from the original bridge is described in Appendix E.2.

Results. Across hallucination, corrigibility, and myopia, increasing the support penalty from $\lambda = 0$ to $\lambda = 10$ brings endpoint outputs closer to those decoded from observed activations, at the cost of lower steering strength (Table 8). This trade-off appears for both RMS and optimized straight paths. Allowing curved paths improves both measures on hallucination: at $\lambda = 10$, RMS achieves higher steering strength than optimized straight paths (0.7487 versus 0.7334) and lower output Hellinger distance (0.0086 versus 0.0098). The benefit is less consistent on corrigibility and myopia, where neither path family consistently improves both measures. Tuned mean-difference steering also reaches the highest corrigibility score, but with a larger output distance. These results suggest that support regularization helps control output proximity, while the benefit of curvature depends on the task. We regard these findings as preliminary, since we have not extensively tuned the method for this setting. Detailed comparisons and discussion are given in Appendix E.7.

6 LIMITATIONS AND CONCLUSION

We bring data-driven approaches from manifold learning to activation steering, building on work such as GAGA (Sun et al., 2025). We ground steering in Riemannian geometry: the model's map determines how movement in activation space changes outputs, while observed activations guide where paths should stay. This gives a principled formulation without requiring intermediate centroid targets or a prescribed manifold topology.

Generation and evaluation. Our real-world experiments use open-ended prompts with paired answer texts. We intervene at one prompt position and score steering success across answer tokens, while the metric and output-proximity measure use the next-token distribution. The paired answers also guide optimization, so these results do not establish generalization to unseen prompts (Appendix E.6). Extending steering across generation steps and evaluating complete responses remain open directions, as well as testing these methods in more applied tasks.

Estimating data support. While we show how an idealized support penalty can bound deviation from the manifold in activation and output space (Appendix B.2), in practice, we estimate support from observed activations, and its quality depends on their manifold coverage. We have not established which estimate best keeps decoded paths close to the output distribution. One direction is to estimate the manifold locally and use distance to this estimate to guide paths, without prescribing its global topology.

Computational cost. The LLM implementation approximates pullback length using differences between model outputs, avoiding an explicit full Jacobian (Appendix C.2.2). Repeated model evaluations still make path optimization more expensive than applying a fixed steering vector. A smaller surrogate map, mimicking the actual neural network, could reduce this cost. This would require checking that it preserves the model's output geometry and that its paths retain their quality when evaluated through the actual model.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing refinement, including grammar and phrasing and bringing the draft to the page limit; identifying potentially relevant related work; checking and correcting bibliography entries; editing figures; debugging and reviewing code implementations. We did not use generative AI tools to formulate hypotheses or translate content.

- Generating datasets: not used.
- Developing theoretical models or conceptual frameworks: not used.
- Formulating mathematical claims: not used.
- Providing critical ingredients for proofs: used (AI suggestions for critical ingredients and missing assumptions).
- Writing proofs: used (to format the proofs and improve notation readability).
- Designing or giving feedback on methodology or experiments: used (analyzing experiments and writing).
- Implementing methods: used (mostly to replicate methods and baselines).
- Cleaning or reformatting datasets: not applicable.
- Qualitative or thematic data analysis: not used.
- Interpreting results: used.

All AI-assisted outputs were reviewed and verified by the authors. In particular, sources identified through generative AI were independently checked, and AI-assisted code and debugging suggestions were reviewed and tested for correctness. We take full responsibility for the final content of this work, including all text, claims, code, evaluations, and other artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies activation steering using synthetic tasks and existing language-model benchmarks. Steering can encourage desirable behaviors, but it can also amplify undesirable ones. Our experiments include steering toward hallucination-aligned answers as a controlled test of behavioral change, rather than a deployment objective. Proximity to observed output distributions does not establish factual accuracy, fairness, or safety: the reference distributions may themselves reflect errors or biases. Our results should therefore be interpreted as evidence about steering and distributional proximity, rather than guarantees of trustworthy generation.

REPRODUCIBILITY STATEMENT

The appendices describe the mathematical formulation, dataset construction and selection, model and intervention settings, baseline definitions, optimization hyperparameters, and evaluation procedures. We specify the numerical approximations used to optimize and evaluate paths, and document reference-path construction and the selection of illustrative examples. Saved experiment configurations and evaluation outputs support regenerating the reported tables and figures. We distinguish descriptive variation across prompts or endpoint pairs from variation across independent optimization runs, and state where the same examples are used for optimization and evaluation.

REFERENCES

Giovanni S. Alberti, Johannes Hertrich, Matteo Santacesaria, and Silvia Sciutto. Manifold learning by mixture models of VAEs for inverse problems. *Journal of Machine Learning Research*, 25 (202):1–35, 2024.

Shun-ichi Amari. *Information Geometry and Its Applications*, volume 194 of *Applied Mathematical Sciences*. Springer, 2016.

Alessandro Benfenati and Alessio Marta. A singular Riemannian geometry approach to Deep Neural Networks I. Theoretical foundations. *Neural Networks*, 158:331–343, 2023a.

Alessandro Benfenati and Alessio Marta. A singular Riemannian geometry approach to Deep Neural Networks II. Reconstruction of 1-D equivalence classes. *Neural Networks*, 158:344–358, 2023b.

Alessandro Benfenati, Alfio Ferrara, Alessio Marta, Davide Riva, and Elisabetta Rocchetti. Unveiling transformer perception by exploring input manifolds. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 21760–21778. Curran Associates, Inc., 2025. doi: 10.52202/085713-0734.

Usha Bhalla, Thomas Fel, Can Rager, Sheridan Feucht, Tal Haklay, Daniel Wurgaft, Siddharth Boppana, Matthew Kowal, Vasudev Shyam, Jack Merullo, Atticus Geiger, and Ekdeep Singh Lubana. Do sparse autoencoders capture concept manifolds? 2026. URL https://arxiv.org/abs/2604.28119.

Kirill Bunin, Dmitry Bylinkin, Vladimir Aletov, Daniil Medyakov, Vladimir Solodkin, and Aleksandr Beznosikov. Controlling refusal behavior of llms via stiefel-constrained rotation steering. *arXiv preprint arXiv:2608.30986*, 2026.

Manfredo Perdigão do Carmo. *Differential geometry of curves and surfaces. With a new preface by the author*. Mineola, NY: Dover Publications, revised and updated 2nd edition of the 1976 edition published by Prentice-Hall edition, 2016. ISBN 978-0-486-80699-0.

Margaret Duff, Neill D. F. Campbell, and Matthias Joachim Ehrhardt. Regularising inverse problems with generative machine learning models. *Journal of Mathematical Imaging and Vision*, 66:37–56, 2024.

Jacob Dunefsky and Arman Cohan. One-shot optimized steering vectors mediate safety-relevant behaviors in LLMs. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=teW4nIZ1gy.

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

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Wes Gurnee, Emmanuel Ameisen, Isaac Kauvar, Julius Tarng, Adam Pearce, Chris Olah, and Joshua Batson. When models manipulate manifolds: The geometry of a counting task. *Transformer Circuits Thread*, 2025. URL https://transformer-circuits.pub/2025/linebreaks/index.html.

Subhash Kantamneni and Max Tegmark. Language models use trigonometry to do addition. *arXiv preprint arXiv:2502.00873*, 2025.

Dhruva Karkada, Daniel J. Korchinski, Andres Nava, Matthieu Wyart, and Yasaman Bahri. Symmetries in language statistics shape the geometry of model representations. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/63405.

John M. Lee. *Introduction to Riemannian manifolds*, volume 176 of *Grad. Texts Math.* Cham: Springer, 2nd edition edition, 2018. ISBN 978-3-319-91754-2; 978-3-319-91755-9. doi: 10.1007/978-3-319-91755-9.

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989. doi: 10.1007/BF01589116.

Ashwin Narayan, Bonnie Berger, and Hyunghoon Cho. Assessing single-cell transcriptomic variability through density-preserving data visualization. *Nature Biotechnology*, 39:765–774, 2021. doi: 10.1038/s41587-020-00801-7.

Narmeen Fatimah Oozeer, Luke Marks, Fazl Barez, and Amir Abdullah. Beyond linear steering: Unified multi-attribute control for language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 23513–23557. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-emnlp.1278. URL https://aclanthology.org/2025.findings-emnlp.1278/.

Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39643–39666, 2024. URL https://proceedings.mlr.press/v235/park24c.html.

Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. The geometry of categorical and hierarchical concepts in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=bVTM2QKYuA.

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.

Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.828. URL https://aclanthology.org/2024.acl-long.828/.

Or David Shafran, Shaked Ronen, Omri Fahn, Shauli Ravfogel, Atticus Geiger, and Mor Geva. From directions to regions: Decomposing activations in language models via local geometry. In *Forty-third International Conference on Machine Learning*, 2026.

Hang Shao, Abhishek Kumar, and P. Thomas Fletcher. The Riemannian geometry of deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 315–323, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018_workshops/papers/w10/Shao_The_Riemannian_Geometry_CVPR_2018_paper.pdf.

Barak Sober and David Levin. Manifold approximation by moving least-squares projection (MMLS). *Constructive Approximation*, 52(3):433–478, 2020. doi: 10.1007/s00365-019-09489-8.

Xingzhi Sun, Danqi Liao, Kincaid MacDonald, Yanlei Zhang, Guillaume Huguet, Guy Wolf, Ian Adelstein, Tim G. J. Rudner, and Smita Krishnaswamy. Geometry-aware generative autoencoders for warped Riemannian metric learning and generative modeling on data manifolds. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 1018–1026. PMLR, 2025. URL https://proceedings.mlr.press/v258/sun25c.html.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Loring W. Tu. *Differential geometry. Connections, curvature, and characteristic classes*, volume 275 of *Grad. Texts Math.* Cham: Springer, 2017. ISBN 978-3-319-55082-4; 978-3-319-85562-2; 978-3-319-55084-8. doi: 10.1007/978-3-319-55084-8.

Minh Hieu Vu and Tan Nguyen. Angular steering: Behavior control via rotation in activation space. *Advances in Neural Information Processing Systems*, 38, 2025.

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.

Zejia You, Chunyuan Deng, and Hanjie Chen. Spherical steering: Geometry-aware activation rotation for language models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.08169.

A Related Work

Activation steering. Activation steering changes model behavior through interventions on internal representations. Contrastive Activation Addition constructs steering directions from differences between labeled examples (Rimsky et al., 2024), while Park et al. (2024) formalize the connections between linear representations, steering, and the choice of inner product. More recent methods account for nonlinear geometry through activation rotations (Vu & Nguyen, 2025; You et al., 2026) or nonlinear feature maps (Raval et al., 2026). Our work studies how to construct an entire steering path that accounts for output changes while remaining close to observed activations.

Geometry of learned representations. Neural representations can organize concepts into curved structures, including circles for weekdays and months (Engels et al., 2025) and helices for numerical computation (Kantamneni & Tegmark, 2025). These studies use interventions to examine how the structures contribute to computation. Work on their origins further connects representation geometry to statistical structure in the training data (Karkada et al., 2026). These findings motivate steering methods that use the observed geometry without requiring its form to be specified in advance.

Manifold steering. Wurgaft et al. (2026) connect activation geometry to the output trajectories induced by steering. They construct manifold paths using splines through concept centroids, recover activation paths by matching prescribed output trajectories, and formulate steering through flat, density-based, and pullback metrics. We build on this geometric view. Our objective combines a pullback metric through the frozen network with a separate activation-support penalty, allowing us to optimize paths without intermediate centroid targets or a prescribed output trajectory. Our synthetic experiments also distinguish accounting for output geometry from remaining near the activation surface.

Manifold learning and data support. Manifold-learning methods estimate structure from samples through local reconstruction (Sober & Levin, 2020) or learned generative charts (Alberti et al., 2024). GAGA learns geometry from point clouds and modifies a learned pullback metric to discourage paths from leaving the data manifold (Sun et al., 2025). Our work brings this data-dependent perspective to activation steering: in our LLM experiments, the metric comes from the frozen network's output map, while a separate penalty encourages proximity to observed activations. This soft treatment of support also connects to generative regularization for inverse problems, where penalties relax strict constraints to a learned manifold (Duff et al., 2024).

Riemannian geometry and neural-network maps. Our formulation draws on the classical treatment of metrics and geodesics (Tu, 2017; Lee, 2018) and the geometry of probability distributions (Amari, 2016). Shao et al. (2018) use decoder-induced metrics to compute geodesics in generative models. Benfenati & Marta (2023a;b) develop singular pullback geometry for neural networks and algorithms for exploring inputs within and across classes sharing the same output. Benfenati et al. (2025) extend this approach to Transformers. These works provide the geometric foundations for our use of the network's output map to guide steering paths.

B MATHEMATICAL DETAILS

B.1 MANIFOLD-CONSTRAINED GEODESICS

Let the ambient activation space be $\mathbb{R}^n$, equipped with a smooth Riemannian metric g: each $g(x)$ is a symmetric positive-definite matrix. Let $\mathcal{M} \subset \mathbb{R}^n$ be a smooth embedded data manifold of dimension $d \ll n$, closed as a subset of $\mathbb{R}^n$, with the metric induced by g. Given two endpoints $x_0, x_1 \in \mathcal{M}$ we seek the shortest path on $\mathcal{M}$ connecting them. If $\mathcal{M}$ is compact (more generally, complete) and x_0, x_1 lie in the same connected component, a minimizing geodesic exists by the Hopf–Rinow theorem Lee (2018); Tu (2017); examples are the spheres S^d, the tori T^d and their connected sums, e.g. a genus-two surface obtained by gluing two 2D tori. Compactness does not imply uniqueness – for example the antipodal points of a sphere are joined by infinitely many minimizing geodesics – but uniqueness holds when x_0, x_1 lie in a geodesically convex subset of $\mathcal{M}$. Throughout this theoretical discussion, we assume that $\mathcal{M}$ is connected and complete in its induced metric. We denote the fixed-endpoint paths in the ambient space by

$$\Gamma_{x_0,x_1} = \big\{\gamma \in \mathcal{C}^0([0,1];\mathbb{R}^n) \text{ Lipschitz and such that } \gamma(0) = x_0,\ \gamma(1) = x_1\big\}. \tag{7}$$

This definition does not require the path to remain on $\mathcal{M}$. For a Lipschitz path, derivatives are defined almost everywhere. Its metric speed is

$$s_\gamma(t) = \sqrt{\dot\gamma(t)^\top g(\gamma(t))\dot\gamma(t)}, \qquad L[\gamma] = \int_0^1 s_\gamma(t)\,dt. \tag{8}$$

The Riemannian distance $d_{\mathcal{M}}(x_0, x_1)$ is the infimum of the lengths of piecewise smooth curves in $\mathcal{M}$ joining x_0 and x_1. Under our assumptions $d_{\mathcal{M}}(x_0, x_1)$ equals the length of any minimizing geodesic from x_0 to x_1.

Finding the geodesic $\gamma^\star$ connecting two given points can be formulated as the constrained minimization problem

$$\gamma^\star \in \operatorname*{arg\,min}_{\gamma\in\Gamma_{x_0,x_1}} L[\gamma] \qquad \text{subject to} \qquad \gamma(t) \in \mathcal{M}\ \ \forall t \in [0,1]. \tag{9}$$

Introducing the on-manifold indicator $\iota_{\mathcal{M}}(x) = 0$ if $x \in \mathcal{M}$ and $+\infty$ otherwise, equation 9 admits the equivalent formulation (Alberti et al., 2024)

$$\gamma^\star \in \operatorname*{arg\,min}_{\gamma\in\Gamma_{x_0,x_1}} L[\gamma] + \int_0^1 \iota_{\mathcal{M}}\big(\gamma(t)\big)dt \tag{10}$$

Since $\mathcal{M}$ is closed and γ is continuous, any departure from $\mathcal{M}$ persists over an interval of positive length. The indicator integral is therefore zero for an on-manifold path and infinite otherwise.

In practice, we do not know the manifold and cannot evaluate its indicator. We instead use a finite continuous support penalty $r\colon \mathbb{R}^n \to [0,\infty)$ (Duff et al., 2024). In the idealized formulation, r vanishes exactly on $\mathcal{M}$ and increases with distance from it; a penalty estimated from data need only be small near observed activations. We define the average penalty along the parameter interval by

$$R[\gamma] = \int_0^1 r(\gamma(t))\,dt. \tag{11}$$

Unlike length, R depends on how quickly the path is traversed. Without a speed restriction, a path can spend most of the interval on $\mathcal{M}$ and cross unsupported regions arbitrarily quickly, reducing R without changing its length or its largest departure from the manifold.

For the theoretical length formulation, we therefore require constant metric speed. On the interval $[0,1]$, this means $s_\gamma(t) = L[\gamma]$ almost everywhere. We consider the relaxation

$$\begin{aligned} \gamma^\star \in \operatorname*{arg\,min}_{\gamma\in\Gamma_{x_0,x_1}} & L[\gamma] + \lambda R[\gamma], \\ \text{subject to} \quad & s_\gamma(t) = L[\gamma] \quad \text{for almost every } t \in [0,1], \end{aligned} \tag{12}$$

where $\lambda > 0$. Constant speed prevents the penalty from being reduced by changing the time spent on different parts of the same path. This is an additional assumption for the theoretical problem; uniform sampling in the numerical implementation does not enforce it.

For the hard on-manifold problem, minimizing energy gives the same shortest geometric paths as minimizing length, with energy selecting constant-speed parameterizations do Carmo (2016). The energy is

$$E[\gamma] = \int_0^1 \dot{\gamma}(t)^\top g(\gamma(t))\dot{\gamma}(t)dt. \tag{13}$$

On $[0,1]$, $E[\gamma] \geq L[\gamma]^2$, with equality at constant metric speed. A separate soft formulation uses energy without an explicit speed constraint:

$$\gamma^\star \in \operatorname*{arg\,min}_{\gamma \in \Gamma_{x_0,x_1}} E[\gamma] + \lambda_E R[\gamma], \tag{14}$$

with $\lambda_E > 0$. Energy charges rapid motion through its squared-speed term, but adding R need not yield constant-speed minimizers or the same geometric paths as the penalized length problem. Its numerical evaluation also avoids the square root in the length integrand. The synthetic experiments use this energy formulation, with the discrete scaling specified in Appendix D.3.1.

B.2 On-Off Manifold Regularizing Distance

The path-level support penalty $R[\gamma]$ in equation 12 is intended to keep γ near $\mathcal{M}$. For an idealized distance-to-support regularizer, Lemma B.1 bounds the path's largest deviation from $\mathcal{M}$ under a metric-speed bound and an objective bound. For fixed V and L_0, this upper bound decreases as $\lambda^{-1/2}$ when the penalty weight λ increases. Corollary B.2 transfers the bound to decoded outputs.

Let $d_g(x,y)$ be the infimum of g-lengths over curves joining x to y in the ambient activation space. The idealized regularizer is $r(x) = d_g(x,\mathcal{M}) := \inf_{z\in\mathcal{M}} d_g(x,z)$. We use the metric speed s_γ defined in Equation 8.

Lemma B.1 (Idealized support control)**.** *Suppose $x_0, x_1 \in \mathcal{M}$ and let $\gamma \in \Gamma_{x_0,x_1}$ satisfy $s_\gamma(t) \leq V$ almost everywhere, where $V > 0$ is a fixed bound on metric speed. For $\lambda > 0$, any such curve satisfying $L[\gamma] + \lambda R[\gamma] \leq L_0$ for some $L_0 > 0$ obeys*

$$\max_{t\in[0,1]} d_g\big(\gamma(t),\mathcal{M}\big) \leq \sqrt{\frac{VL_0}{\lambda}}. \tag{15}$$

Proof. Set $u(t) = d_g(\gamma(t),\mathcal{M})$. The distance-to-manifold function $x \mapsto d_g(x,\mathcal{M})$ is 1-Lipschitz with respect to d_g: the triangle inequality, followed by the infimum over $z \in \mathcal{M}$, gives

$$|d_g(x,\mathcal{M}) - d_g(y,\mathcal{M})| \leq d_g(x,y). \tag{16}$$

For $s < t$, the segment of γ between these times is a path from $\gamma(s)$ to $\gamma(t)$. The speed bound consequently gives

$$|u(t) - u(s)| \leq d_g(\gamma(t),\gamma(s)) \leq \int_s^t s_\gamma(v)\,dv \leq V(t-s). \tag{17}$$

Thus u is V-Lipschitz in t and attains a maximum $A = u(t_*)$ for some $t_* \in [0,1]$. If $A = 0$, equation 15 is immediate. Otherwise, because the endpoints lie in $\mathcal{M}$, we have $u(0) = u(1) = 0$. The Lipschitz bound yields $u(t) \geq A - V|t - t_*|$. At the endpoints this gives $t_* \geq A/V$ and $1 - t_* \geq A/V$, and therefore

$$R[\gamma] = \int_0^1 u(t)dt \geq \int_0^1 \max\{0,\ A - V|t-t_*|\}dt = 2\int_0^{A/V} (A - Vv)\,dv = \frac{A^2}{V}. \tag{18}$$

Since $L[\gamma] \geq 0$, the objective comparison implies $R[\gamma] \leq L_0/\lambda$. Combining the two inequalities yields $A^2/V \leq L_0/\lambda$, which proves equation 15. □

Applying the bound to constant-speed paths. Choose a constant-speed comparison path on $\mathcal{M}$ with length at most $L_0 > 0$. Its support penalty is zero. Any constant-speed candidate whose objective is no larger satisfies $s_\gamma(t) = L[\gamma] \leq L_0$ almost everywhere. The lemma therefore applies with $V = L_0$ and gives

$$\max_{t\in[0,1]} d_g(\gamma(t),\mathcal{M}) \leq \frac{L_0}{\sqrt{\lambda}}. \tag{19}$$

The comparison path can be fixed independently of λ, so this bound decreases as the penalty weight increases. The conclusion depends on the stated speed and objective bounds; it is not a property of every path with a small sampled penalty.

Corollary B.2 (Decoded-output proximity)**.** *Under the assumptions of Lemma B.1, fix a prompt context c and use p_c and F_c from equation 3. Assume p_c is smooth and strictly positive, and that $g = J_{F_c}^\top J_{F_c} + \epsilon I_n$ with $\epsilon > 0$. We use the convention $H(p,q) = \|\sqrt{p} - \sqrt{q}\|_2/\sqrt{2}$. Then*

$$\max_{t\in[0,1]} \inf_{z\in\mathcal{M}} H\big(p_c(\gamma(t)), p_c(z)\big) \le \sqrt{\frac{VL_0}{\lambda}}. \tag{20}$$

Proof. The Hellinger embedding in equation 3 gives $H(p_c(x), p_c(z)) = \|F_c(x) - F_c(z)\|_2$. For any curve η from x to z, the chain rule and the definition of g give

$$H(p_c(x), p_c(z)) \le \int_0^1 \|J_{F_c}(\eta(t))\dot\eta(t)\|_2 \, dt \le L[\eta]. \tag{21}$$

Taking the infimum over η gives $H(p_c(x), p_c(z)) \le d_g(x,z)$. Taking a second infimum over $z \in \mathcal{M}$ and applying Lemma B.1 proves equation 20. □

These bounds use exact distance to $\mathcal{M}$ and a bound on metric speed along the continuous path. Estimated support penalties and sampled objectives do not automatically satisfy these assumptions.

B.3 Numerical approximation of geodesic curves

We sample a path at $t_i = i/N$, $i = 0, \dots, N$, with fixed endpoints $\gamma_0 = x_0$ and $\gamma_N = x_1$. These uniform parameter values need not give equal distances along the path. The numerical objectives below approximate length or energy and the support penalty, but do not impose the constant-speed constraint in Equation 12.

Write $\Delta_i = \gamma_i - \gamma_{i-1}$ and $\bar\gamma_i = (\gamma_{i-1} + \gamma_i)/2$. A midpoint approximation to length and a mean support penalty give the free-waypoint problem

$$\operatorname*{arg\,min}_{\gamma_1,\dots,\gamma_{N-1}} \sum_{i=1}^N \sqrt{\Delta_i^\top g(\bar\gamma_i)\Delta_i} + \frac{\lambda}{N+1}\sum_{i=0}^N r(\gamma_i). \tag{22}$$

Only the interior waypoints are optimized. The denominator $N+1$ counts all sampled points, including both endpoints.

For the regularized pullback metric $g = J_F^\top J_F + \epsilon I_n$, where $F : \mathbb{R}^n \to \mathbb{R}^k$ and $\epsilon > 0$, it is computationally convenient not to form the full matrix g. The identity

$$\Delta_i^\top g(\bar\gamma_i)\Delta_i = \|J_F(\bar\gamma_i)\Delta_i\|_2^2 + \epsilon\|\Delta_i\|_2^2$$

replaces the matrix contraction with a Jacobian–vector product and the activation-space term. Equation 22 then reads

$$\operatorname*{arg\,min}_{\gamma_1,\dots,\gamma_{N-1}} \sum_{i=1}^N \sqrt{\|J_F(\bar\gamma_i)\Delta_i\|_2^2 + \epsilon\|\Delta_i\|_2^2} + \frac{\lambda}{N+1}\sum_{i=0}^N r(\gamma_i). \tag{23}$$

Similarly, the energy formulation in Equation 14 has the approximation

$$\operatorname*{arg\,min}_{\gamma_1,\dots,\gamma_{N-1}} N\sum_{i=1}^N \big(\|J_F(\bar\gamma_i)\Delta_i\|_2^2 + \epsilon\|\Delta_i\|_2^2\big) + \frac{\lambda_E}{N+1}\sum_{i=0}^N r(\gamma_i). \tag{24}$$

The factor N comes from the time step $1/N$ and is needed to approximate the continuous energy. The synthetic experiments use the unscaled sum and report penalty weights for that convention (Appendix D.3.2). Expressing those objectives in the convention above requires multiplying both their energy and their penalty weight by N.

For LLMs, we optimize the parameters of a continuous neural bridge rather than free waypoints. We approximate length using differences between mapped outputs, with the same activation-space

contribution (Equation 32), and average support at the sampled bridge points (Equation 35). The bridge enforces its endpoints, but not constant metric speed. The existing weekday resolution check evaluates saved bridges on a finer grid (Appendix C.2.2); it checks the sampled length, not the speed assumption or the largest departure between samples. The support bounds above therefore remain conditional theoretical results.

B.4 Estimating the activation-support penalty

We estimate the pointwise penalty $r(x)$ from model activations on a given set of data samples, denoted x_j^{obs}. The path penalty is $R[\gamma] = \int_0^1 r(\gamma(t))\,dt$, as in Equation 4. Below we define the two estimates used in our experiments. These finite-data estimates measure proximity to the observed activations; they are not the exact distance to $\mathcal{M}$ assumed in Appendix B.2.

B.4.1 Synthetic support penalty

The synthetic surfaces lie in a three-dimensional subspace before noise is added. Let μ be the sample mean and let the columns of $P \in \mathbb{R}^{24\times 3}$ be the leading principal-component directions. We write

$$z(x) = P^\top(x-\mu), \qquad e_\perp(x) = (x-\mu) - Pz(x). \tag{25}$$

Here $z(x)$ gives the coordinates within the subspace, while $e_\perp(x)$ is the residual outside it. With $z_j = z(x_j^{\mathrm{obs}})$, define

$$d_{10}(z) = \frac{1}{10}\sum_{j\in\mathcal{N}_{10}(z)} \|z - z_j\|_2, \tag{26}$$

$$r(x) = \sqrt{\|e_\perp(x)\|_2^2 + [d_{10}(z(x)) - \tau]_+^2 + 10^{-30}}. \tag{27}$$

The set $\mathcal{N}_{10}(z)$ contains the ten nearest observed points in these coordinates, and $[a]_+ = \max(a, 0)$. The threshold τ accounts for the spacing between sampled points. The first term penalizes leaving the subspace; the second penalizes distance from observed points within it. The small constant stabilizes the square root. The data split and threshold calibration are specified in Appendix D.2.2.

B.4.2 LLM support penalty

For LLMs, we build a fixed graph whose vertices are the observed activations x_j^{obs}. Edge weights approximate pullback lengths under the metric g_c for a fixed prompt context c. Let $d_G(a, j)$ be the shortest-path distance between vertices a and j in this graph.

To score a new activation x, we select candidate entry vertices $\mathcal{C}(x)$. For each candidate a, define the straight activation segment

$$\eta_{x,a}(s) = (1-s)x + sx_a^{\mathrm{obs}}, \qquad s \in [0, 1]. \tag{28}$$

We estimate its length $\widehat{L}[\eta_{x,a}]$ under g_c using the sampled-length formula in Appendix C.2.2, and define

$$\widehat{d}_{g_c}(x, x_j^{\mathrm{obs}}) = \min_{a\in\mathcal{C}(x)} \left[\widehat{L}[\eta_{x,a}] + d_G(a, j)\right], \tag{29}$$

$$r(x) = \frac{1}{5}\sum_{j\in\mathcal{N}_5^{g_c}(x)} \widehat{d}_{g_c}(x, x_j^{\mathrm{obs}}). \tag{30}$$

The set $\mathcal{N}_5^{g_c}(x)$ contains the five observed activations with the smallest estimated distances. Thus each distance combines a new connection to the graph with a stored distance through it. The graph estimates the penalty; the neural bridge is not required to follow its edges. Graph construction, candidate selection and connection-length sampling are specified in Appendix C.2.4. The age task uses the same construction (Appendix C.7).

C TOY LLM EXPERIMENTS

C.1 TOY LLM DATASETS

Both tasks use the frozen Llama-3.1-8B-Instruct model (Grattafiori et al., 2024). We extract the 4,096-dimensional residual-stream activation at the final prompt token after layer 28. The train/test labels below describe the prompt splits; we do not train the language model. For both tasks, the training activations form the support graph used to optimize paths, and the same training examples supply the references for the reported activation and output KNN distances. These scores therefore measure proximity to the training references, not performance on an independent test set.

C.1.1 DAYS OF THE WEEK

The task asks for a weekday after adding an offset to a starting day. Prompts span 21 templates and seven weekday answer classes. Outputs are grouped into seven weekday-token bins and one bin for the remaining vocabulary.

Example prompt templates. Two templates, shown with example values, are:

> Q: If today is Monday, what day comes two days later?
> A:
>
> Q: Advance the calendar from Monday by two days. What day is it?
> A:

Both prompts have the answer *Wednesday*. The model continues from the answer prefix shown above.

Dataset split and use. There are **1,029 prompts: 823 training and 206 test**. The seven activation centroids are computed from all 1,029 prompts, including the test prompts.

Endpoint pairs. We evaluate all $\binom{7}{2} = 21$ unordered pairs of weekday centroids under a fixed carrier prompt. Our optimizer receives only the two endpoint centroids and the unlabeled reference activations; the spline baseline also uses intermediate centroids and their order. Intervention and optimization details are given in Appendix C.2.

Density-aware visualization. Figure 3 uses densMAP (Narayan et al., 2021) to retain differences in local density. Activation distances come from the saved regularized pullback graph; output distances are Hellinger distances. We embed the 823 training references and seven centroids together, with equal weight and no weekday labels or prescribed ordering. We use 15 neighbors, a minimum distance of 0.1, 500 epochs, and density weight 2. The displayed seed is 0. Across seeds 0, 1, and 2, the Spearman correlation between mean five-neighbor distances before and after projection is 0.809–0.831 for activations and 0.770–0.781 for outputs. These checks use reference points only and exclude each point from its own neighbors. The projection retains local density differences imperfectly; support is therefore evaluated using distances before projection.

The centroids' mean five-neighbor distances fall within the 41st–75th percentiles of reference-point distances in activation space and the 29th–63rd percentiles in output space. Higher percentiles indicate less densely sampled regions. These scores place the centroids within the central range of the reference data, rather than among the sparsest points.

Activation centroids are attached to their 16 nearest references by endpoint chords in $[\sqrt{p_c(x)}/\sqrt{2}, \sqrt{\epsilon}x]$, followed by saved graph paths. Distances between centroids use the shortest connection through a reference; reference-to-reference distances remain fixed. These display attachments approximate the integrated attachments used for evaluation, so star positions alone should not be used to judge centroid support. The graph uses the fixed carrier prompt, while natural output references retain their own prompts and centroid outputs use the carrier. The two panels are separate views, not an isometry test.

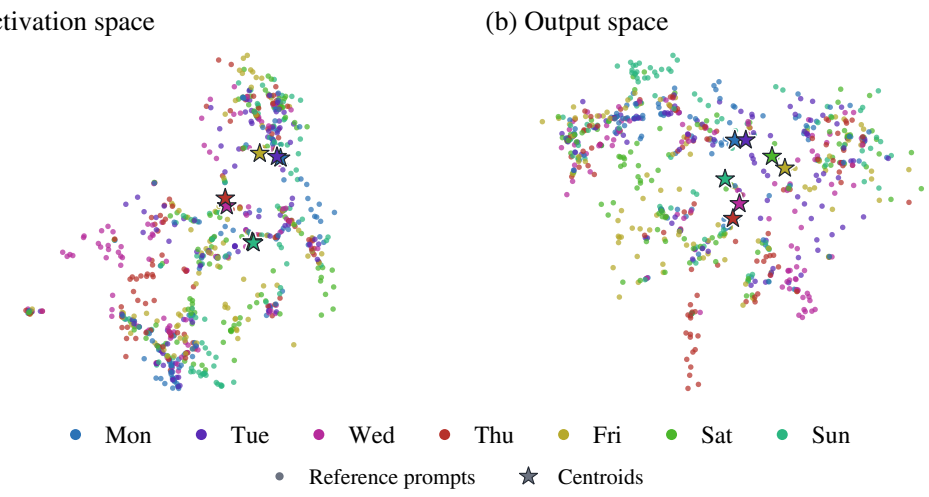

Figure 3: **Density-aware views of weekday representations.** Dots show reference activations (left) and their natural outputs (right); stars show weekday centroids and their decoded outputs. Colors indicate weekdays. We use densMAP (Narayan et al., 2021) to preserve differences in local density without using weekday labels to arrange the points. Full-space KNN measurements indicate that the centroids lie in fairly dense regions, but not the densest: our paths reach points with lower KNN distances on average (Appendix Figure 4).

C.1.2 AGE ARITHMETIC

The task asks for Bob's age given Alice's age and their age difference. Answers range from 10 to 100, giving 91 age-token bins and one bin for the remaining vocabulary. Ages have a sequential ordering.

Example prompt templates. Two templates, shown with example values, are:

> Q: If Alice is 71 years old and Bob is 2 years older, how old is Bob?
> A: Bob is

> 85 years old (Alice) plus 7 years = (Bob)
> A:

The answers are *73* and *92*, respectively. As above, each prompt ends at its answer prefix.

Dataset split and use. There are **1,100 prompts: 1,000 training and 100 test**. The 91 activation centroids are computed from all 1,100 prompts, including the test prompts.

Endpoint pairs. We select 21 of the 4,095 unordered centroid pairs uniformly without replacement, using NumPy PCG64 with seed 0 before evaluating the paths. As for weekdays, our optimizer receives only the two endpoint centroids and the unlabeled reference activations. The model extraction, optimization and evaluation details are given in Appendix C.7.

C.2 WEEKDAY METHOD AND EVALUATION

We instantiate the objective in Equation 4 with the actual network's pullback length and a graph-based activation-support penalty. Section 3 gives the support-constrained motivation; here we specify the neural bridge, the numerical loss and the reference distances used in the reported runs. The dataset and task are defined in Section C.1.1.

C.2.1 INTERVENTION AND BASELINES

We use the reference set and activation extraction described in Appendix C.1.1. Every method replaces the full activation, while earlier-token states retain one fixed prompt, indexed 417. Outputs

are decoded without conditioning on a weekday. This is a shared full-activation intervention, not an exact replication of the original carrier-residual intervention of Wurgaft et al. (2026).

The spline steering baseline fits a periodic cubic spline through all seven full activation centroids, with angles obtained from the canonical PCA-64 construction and the shorter angular arc between endpoints. Linear steering uses their full-activation chord. We train a separate endpoint-constrained bridge for each endpoint pair: two hidden layers of width 32, seed 0, and ten Adam updates with learning rate 10^{-3} and zero weight decay. The conditioning scalar is fixed to two for every pair; it does not encode weekday order. These fixed-budget runs are not certified converged solutions.

C.2.2 Output metric and sampled path length

Using the construction in Equation 2, for the frozen network under prompt c we have

$$F_c(x) = \frac{\sqrt{p_c(x)}}{\sqrt{2}}, \qquad A_c(x) = [F_c(x), \sqrt{\epsilon}\, x], \qquad g_c(x) = J_{F_c}(x)^\top J_{F_c}(x) + \epsilon I, \tag{31}$$

with $\epsilon = 3.1867797952145338 \times 10^{-6}$. The fixed positive floor keeps changes in activation directions that do not affect these eight output bins visible. We use $L[\gamma]$ with $g = g_c$. The target is length, $L[\gamma] = \int_0^1 \sqrt{\dot\gamma^\top g_c(\gamma)\dot\gamma}\, dt$, not squared energy. With $\gamma_i = \gamma(i/50)$, its polygonal approximation is

$$\widehat{L}[\gamma] = \sum_{i=0}^{49} \sqrt{\|F_c(\gamma_{i+1}) - F_c(\gamma_i)\|_2^2 + \epsilon\|\gamma_{i+1} - \gamma_i\|_2^2}. \tag{32}$$

The segment lengths are summed, not averaged. During path optimization, the language model stays frozen.

Computational cost. To reduce the cost of repeated path evaluations through the frozen language model, we use the polygonal approximation in Equation 32. Differences between mapped waypoints require only first derivatives (Shao et al., 2018), allowing ordinary back-propagation and activation checkpointing without differentiating through sampled Jacobian–vector products. The neural bridge defined in Appendix C.2.3 remains continuous; only its loss is sampled. We minimize length to match the shortest-path target, rather than the squared-energy objective used in the synthetic experiments.

Resolution check. For a fixed piecewise-smooth mapped curve, the polygonal sum converges to its length as the mesh is refined. At finite resolution, chords can miss bends or excursions between samples. For the 21 saved activation-only bridges, evaluating the same networks at 201 rather than 51 time points increases length by a median 0.0088% and a maximum 0.0323%, relative to the 201-point value. This checks the resolution of the saved paths; it does not establish convergence after retraining on a finer mesh, bound the remaining error, or show equivalence to energy optimization.

C.2.3 Neural bridge: the activation path we optimize

For each endpoint pair, we train a small neural network with parameters ψ to produce the continuous activation path

$$\begin{aligned} \gamma_\psi(t) &= (1-t)x_0 + tx_1 + 4t(1-t)\, B_\psi(x_0, x_1, t), \\ B_\psi(x_0, x_1, t) &= M_\psi([E_{0,\psi}(x_0), E_{1,\psi}(x_1), t, I_\psi(2)])\,. \end{aligned} \tag{33}$$

Here $x_0, x_1 \in \mathbb{R}^{4096}$ are the fixed source and target activations, $t \in [0,1]$ is the path parameter, and B_ψ is a learned 4096-dimensional correction to the straight line. The factor $4t(1-t)$ vanishes at both ends, so the endpoint constraints hold for every ψ; we do not enforce them with another loss. The two separate endpoint networks $E_{0,\psi}$ and $E_{1,\psi}$ each have widths $4096 \to 32 \to 32 \to 32$. The correction network M_ψ has widths $66 \to 32 \to 32 \to 4096$. These networks use ReLU after each of their first two affine layers, with no output activation. The scalar network I_ψ has widths $1 \to 1 \to 1$, with one intervening ReLU. Its input is always 2, not the weekday pair index. The 66 inputs to M_ψ are the two 32-dimensional endpoint embeddings, the unembedded scalar t, and the one-dimensional conditioning embedding.

All these small-network parameters belong to ψ (404,804 trainable parameters); the language-model parameters do not. Each pair gets a separate bridge, initialized with seed 0 and the default PyTorch

linear-layer initialization. The initial path is a straight line plus a random correction, not an exactly straight path, because the correction network is not zero-initialized. The bridge receives no intermediate centroid targets or weekday ordering. It is continuous in t, but its ReLU parameterization is not everywhere differentiable.

C.2.4 ACTIVATION-SUPPORT GRAPH AND QUERY CONNECTIONS

We implement the penalty defined in Appendix B.4.2.

The fixed reference graph. We build a k-nearest-neighbor graph of the reference activations, with edge weights given by approximate pullback lengths. We estimate distances between references by finding the path through this graph with the smallest sum of edge weights. The graph is built once and reused for all neural bridges.

Specifically, let x_j^{obs}, $j = 1, \ldots, 823$, denote the reference activations. For each reference, we select its 20 nearest neighbors using straight-line distances between their mapped coordinates $A_c(x_j^{\mathrm{obs}})$. We connect two references if either selects the other, and allow travel in both directions. Each edge weight approximates the pullback length of the straight activation segment joining its endpoints, using Equation 32 with eight subsegments. We store the resulting shortest-path distances $d_G(a, j)$ between reference points a and j. The references, mapped coordinates, edge weights and stored distances stay fixed during optimization; we do not rebuild the graph at each waypoint.

Connecting a current waypoint to that graph. For each query activation $x = \gamma_\psi(t_i)$, we first select a set $\mathcal{C}(x)$ of candidate reference vertices by the chord distances $\|A_c(x) - A_c(x_a^{\mathrm{obs}})\|_2$. We then calculate the cost of connecting x to each candidate x_a^{obs}. Let $\eta_{x,a}$ be the straight activation segment defined in Equation 28. With M subsegments, its sampled points are

$$u_\ell = (1 - \ell/M)x + (\ell/M)x_a^{\mathrm{obs}}, \qquad \ell = 0, \ldots, M. \tag{34}$$

We compute the connection length $\widehat{L}[\eta_{x,a}]$ using Equation 32: replace the bridge waypoints γ_i by these connection samples u_ℓ, and the 50 bridge segments by M subsegments (sum over $\ell = 0, \ldots, M-1$). Thus the same length calculation is applied to a different path and sampling resolution. Computing this connection length requires decoding points along the waypoint-to-reference segment through the frozen network; it is not just the chord distance used to select candidates. We combine this new connection cost with the stored graph distances as in Equation 29. The sampled path penalty is

$$\widehat{R}[\gamma] = \frac{1}{51} \sum_{i=0}^{50} r(\gamma_i), \qquad \gamma_i = \gamma(i/50), \tag{35}$$

where r is the five-neighbor mean in Equation 30. Thus $\widehat{R}[\gamma]$ is the sampled approximation to $R[\gamma]$ using 51 waypoints. In Equation 29, the candidate index a is an entry point into the graph; the reference index j is the destination whose distance is being scored. They need not be the same vertex. One connection length is reused for all 823 destinations through the lookup $d_G(a, j)$, rather than solving a new continuous geodesic to each destination. These connections define the support penalty; they do not force the neural bridge to follow graph edges or pass through the reference vertices.

How often these lengths are recalculated. During every optimizer update, we evaluate the current bridge at all 51 fixed times $t_i = i/50$, including the endpoints. At each waypoint, we select eight candidates and recompute their connection lengths with $M = 4$ subsegments (five sampled points per connection). This is $51 \times 8 = 408$ connection lengths per full update, or 4,080 over the ten training updates for one endpoint pair, excluding diagnostic passes. The waypoint positions depend on ψ, so candidates, connection lengths and the final five nearest references can change after an update. Even the fixed endpoint queries are reevaluated by the implementation. Only the reference graph and d_G are cached, not the moving-waypoint connections. The connection's local samples u_ℓ are auxiliary activation points; they are not extra time samples on the neural bridge.

Final proximity evaluation scores all 51 waypoints of every method again, using sixteen candidates per waypoint and $M = 8$ subsegments (nine points per connection). This gives 816 connection lengths per evaluated path, with no optimizer update. Initial and final component-gradient diagnostics also evaluate connections, separately from training. The finer evaluation does not make

the shared graph an independent test. No activation PCA or hinge is used. The penalty averages uniformly in t, not in arc length, and therefore is not invariant to a change in path parameterization.

C.2.5 NEURAL-BRIDGE OPTIMIZATION

The sampled version of Equation 4 minimizes $\widehat{L}[\gamma_\psi] + 10\widehat{R}[\gamma_\psi]$ over ψ, with no additional output-support penalty. For each of ten Adam updates (learning rate 10^{-3}, zero weight decay), the code processes the 50 adjacent waypoint segments in overlapping two-waypoint microbatches and accumulates their gradients before making one optimizer step. Each of the 50 path-length segments is counted once. To avoid counting support twice at a shared waypoint, the first 49 microbatches score only their left endpoint's support, and the last scores both endpoints. Consequently all 51 support values contribute once, each with weight $1/51$. There is no optimizer step between these microbatches, and no learning-rate scheduler or gradient clipping.

Back-propagation differentiates the sampled connection lengths through their activation points, through the frozen network's input-output map, and then through γ_ψ into the bridge parameters. Language-model weights, prompt context, reference data, graph distances and ϵ are not updated. Candidate indices are selected without gradients; the minimum and five-neighbor average propagate gradients through the currently selected distance values, not through the discrete identities. Neighbor changes can therefore make the objective nonsmooth. This polygonal calculation needs first-order network derivatives, not an explicit Jacobian matrix or a second-order derivative through a sampled Jacobian-vector speed. Activation checkpointing may repeat network forward calculations during back-propagation without adding optimizer updates. We also save the trained curve at 201 times for a denser path-length diagnostic; those points are not extra training waypoints or extra samples in the primary k-NN evaluation.

C.2.6 TRAJECTORY EVALUATION

Why also evaluate decoded outputs? The optimized variables are the neural bridge parameters and the activation path; the output path is induced by the frozen network. If an activation path lay exactly in $\mathcal{M}$, its outputs would lie in the image $p_c(\mathcal{M})$. Our finite graph penalty does not establish that premise. Moreover, natural output references retain their original prompts, whereas the activation graph decodes its references under prompt 417. These two reference clouds are not exactly one another's fixed-context image. We therefore measure output proximity rather than infer it from activation proximity.

We define both proximity scores for one path γ, sampled at $\gamma_i = \gamma(i/N)$, including its endpoints. Observed activations are x_j^{obs}, and p_j^{obs} are the model's outputs on the corresponding data samples. Each p_j^{obs} retains its original prompt context; it need not equal the fixed-context decode $p_c(x_j^{\mathrm{obs}})$.

Activation pullback distance. At each path point, we average its graph-approximated pullback distance to the five nearest observed activations. We then average over the path:

$$S^{\mathrm{act}}[\gamma] = \frac{1}{N+1}\sum_{i=0}^{N}\frac{1}{5}\sum_{j\in\mathcal{N}_5^{\mathrm{act}}(\gamma_i)}\widehat{d}_{g_c}(\gamma_i, x_j^{\mathrm{obs}}). \tag{36}$$

Here $\mathcal{N}_5^{\mathrm{act}}(\gamma_i)$ contains the five closest observed activations under $\widehat{d}_{g_c}$, defined in Equation 29. Lower values mean that the path stays closer to the activation data. This uses the same type of distance as the support penalty, with finer graph connections during evaluation (Appendix C.2.4).

Output Hellinger distance. We decode each path point under the fixed prompt c. We average its Hellinger distance to the five nearest observed output distributions, then average over the path:

$$S^{\mathrm{out}}[\gamma] = \frac{1}{N+1}\sum_{i=0}^{N}\frac{1}{5}\sum_{j\in\mathcal{N}_5^{\mathrm{out}}(\gamma_i)}H\big(p_c(\gamma_i), p_j^{\mathrm{obs}}\big). \tag{37}$$

Here $\mathcal{N}_5^{\mathrm{out}}(\gamma_i)$ contains the five observed outputs closest to $p_c(\gamma_i)$ under Hellinger distance $H(p,q) = \|\sqrt{p} - \sqrt{q}\|_2/\sqrt{2}$. These neighbors are selected in output space, independently of the activation graph. Lower values mean that intermediate outputs stay closer to the observed distributions. This output-to-data distance is not part of the main training objective.

Sampling and aggregation. Both tasks use $N = 50$, giving 51 path points, including endpoints. Distances use all output bins without PCA: eight bins for weekdays and 92 for ages. The observed data and endpoint activations are shared across methods. The same training examples supply the support references and the proximity evaluation; these scores do not constitute an independent held-out test. For each method, we average the path scores over the 21 endpoint pairs. The reported standard error is the sample standard deviation of these 21 scores divided by $\sqrt{21}$. It describes pair spread; paths share endpoints, prompts and reference data. Neither score is multiplied by a training weight. The scores are the two separate columns in Table 2.

C.3 Activation-support weight ablation

Table 3: Monday-to-Thursday activation-support ablation. The actual-network map, initialization, seed and ten-update budget are fixed; the length coefficient is one. Only the activation-support weight varies. Lower trajectory-averaged mean-5-NN distance is better. One prompt and pair, so no independent-run standard error is available.

Method	Output Hellinger	Activation pullback
Linear	0.09350	0.10622
Spline steering	0.09350	0.10325
RMS, $\lambda = 0$	0.09407	0.10845
RMS, $\lambda = 1$	0.09113	0.10160
RMS, $\lambda = 3$	0.09252	0.09685
RMS, $\lambda = 10$	**0.08965**	**0.08073**

All three tested positive activation weights beat linear and spline on both proximity means in this pilot; length alone does not. Weight 10 has the lowest values among these three and is frozen for the 21-pair comparison. This is exploratory weight selection on one pair, not an optimality result. The all-pair comparison includes this selection pair.

For Wednesday-to-Friday, the sole output-distance exception to the 21-pair baseline wins, RMS gives 0.100565, versus 0.100265 for linear and 0.089195 for the spline. Its activation distance remains lower: 0.095523, versus 0.109947 and 0.099830, respectively. An all-pair length-only control and a no-length control have not been run. Full-vocabulary top-five diagnostics from earlier configurations must not be attributed to this activation-only batch.

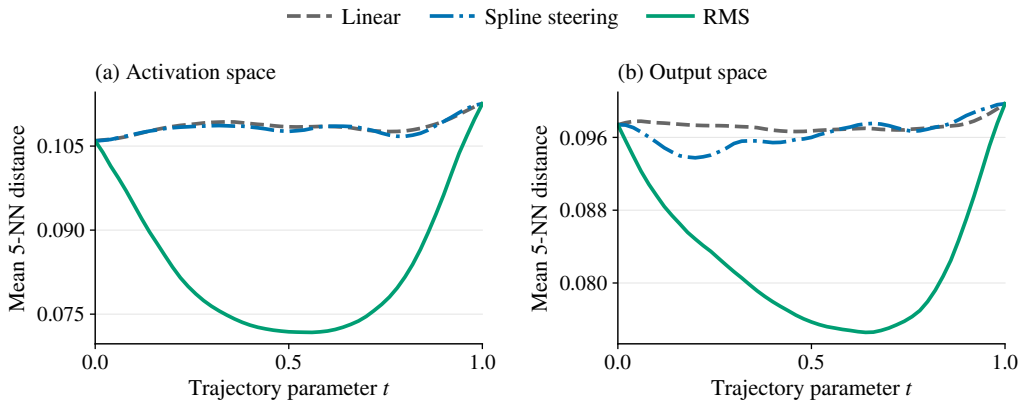

Figure 4: Weekday paths stay closer to the reference data between their endpoints. The curves show average distance along the paths in activation space (left) and output space (right). Lower curves mean closer paths; our method is shown in green.

C.4 WEEKDAY PEAKS AND PATH VISUALIZATIONS

Interior peaks. For a weekday d other than the two endpoints, let $p_d(t)$ be its probability in $p_c(\gamma(t))$. We locate its first sampled maximum and count an interior peak when that maximum occurs between the endpoints:

$$t_d^\star = \min \arg\max_{t_i} p_d(t_i), \qquad 0 < t_d^\star < 1. \tag{38}$$

The minimum selects the earliest time if several samples have the same maximum. We use raw probabilities on the saved 51-point grid and impose no minimum peak size. In the reported paths, every counted peak strictly exceeds that day's probabilities at both endpoints.

Dominant peaks. An interior peak is dominant when its weekday is more probable than every other weekday at that time:

$$p_d(t_d^\star) > \max_{e \neq d} p_e(t_d^\star), \tag{39}$$

where e ranges over the other weekdays. This compares weekday bins, not the full vocabulary, and ties do not count as dominant. Thus an interior peak measures whether a concept becomes more likely along the path; dominance additionally requires it to be the preferred weekday.

Counting across paths. We consider all five non-endpoint weekdays for each of the 21 endpoint pairs, giving 105 pair–day opportunities. We count interior peaks and dominant interior peaks separately. Pooling days from both calendar directions does not establish one ordered path. RMS produces more interior peaks than either baseline, but the spline's peaks are larger and more often dominant (Table 4).

Table 4: Exploratory interior-peak counts. **A:** All 21 pairs. **B:** Monday-to-Thursday only, with five rather than 105 opportunities. RMS uses activation support with $\lambda = 10$. Every non-endpoint day is considered, regardless of path direction. Dominance means exceeding all other weekdays at that day's own interior peak. Median increase is measured above the larger endpoint probability, in percentage points, among detected interior peaks. Counts do not impose ordering or a minimum amplitude.

Method	Interior peaks	Dominant interior peaks	Median increase
A. All 21 pairs			
Linear	18/105	3/105	0.033
Spline steering	21/105	15/105	7.164
RMS	55/105	7/105	0.722
B. Monday-to-Thursday only			
Linear	2/5	0/5	0.292
Spline steering	2/5	1/5	6.434
RMS	3/5	0/5	3.366

A stricter check asks whether at least one nonempty clockwise or anticlockwise path has all intermediate peaks inside the path, with all path peak times strictly ordered. This holds on 8/21 pairs for RMS, 2/21 for linear and 14/21 for the spline. Requiring successive dominant weekday visits along a complete nonempty path gives 0/21, 0/21 and 7/21, respectively. Endpoints are included in both path checks.

Probability masses in the illustrative pair. The Monday-to-Friday example was selected post hoc from the 21 saved pairs to show ordered interior peaks without intermediate centroid targets. Tuesday, Wednesday and Thursday reach 9.51%, 8.52% and 14.04% at $t = 0.38, 0.76, 0.96$, respectively. These exceed their larger endpoint masses by 1.336, 0.775 and 0.057 percentage points. None becomes the most probable weekday; Thursday's late increase is small. Linear peaks at 8.38% for Tuesday and has endpoint maxima for Wednesday and Thursday. RMS also has a Saturday peak, giving four of five non-endpoint days an interior maximum, versus one for linear and two for the spline. The spline follows the opposite calendar arc through Sunday and Saturday, with larger peaks and successive dominance; our example does not show better dominant traversal than the spline.

Figure 5 shows all seven day curves as raw, weekday-conditional and own-peak-scaled masses. Own-peak scaling shows shape, not magnitude or dominance. Conditioning shifts these peaks to

Table 5: Dominance across the same 21 pairs. The middle column conditions on detected interior peaks; the last counts pairs with at least one nonempty calendar path whose days become highest-probability weekdays in order, including endpoints. Other days may intervene between visits. Dominance is among weekday bins, not the full vocabulary. One prompt and seed; descriptive counts without independent-run uncertainty.

Method	Dominant among interior peaks	Pairs with ordered dominant visits
Linear	3/18 (16.7%)	0/21
Spline steering	15/21 (71.4%)	7/21
RMS	7/55 (12.7%)	0/21

$t = 0.32, 0.82, 1.00$, so Thursday is then an endpoint maximum. All-pair tables are unchanged, and the weight ablation in Table 3 remains Monday-to-Thursday.

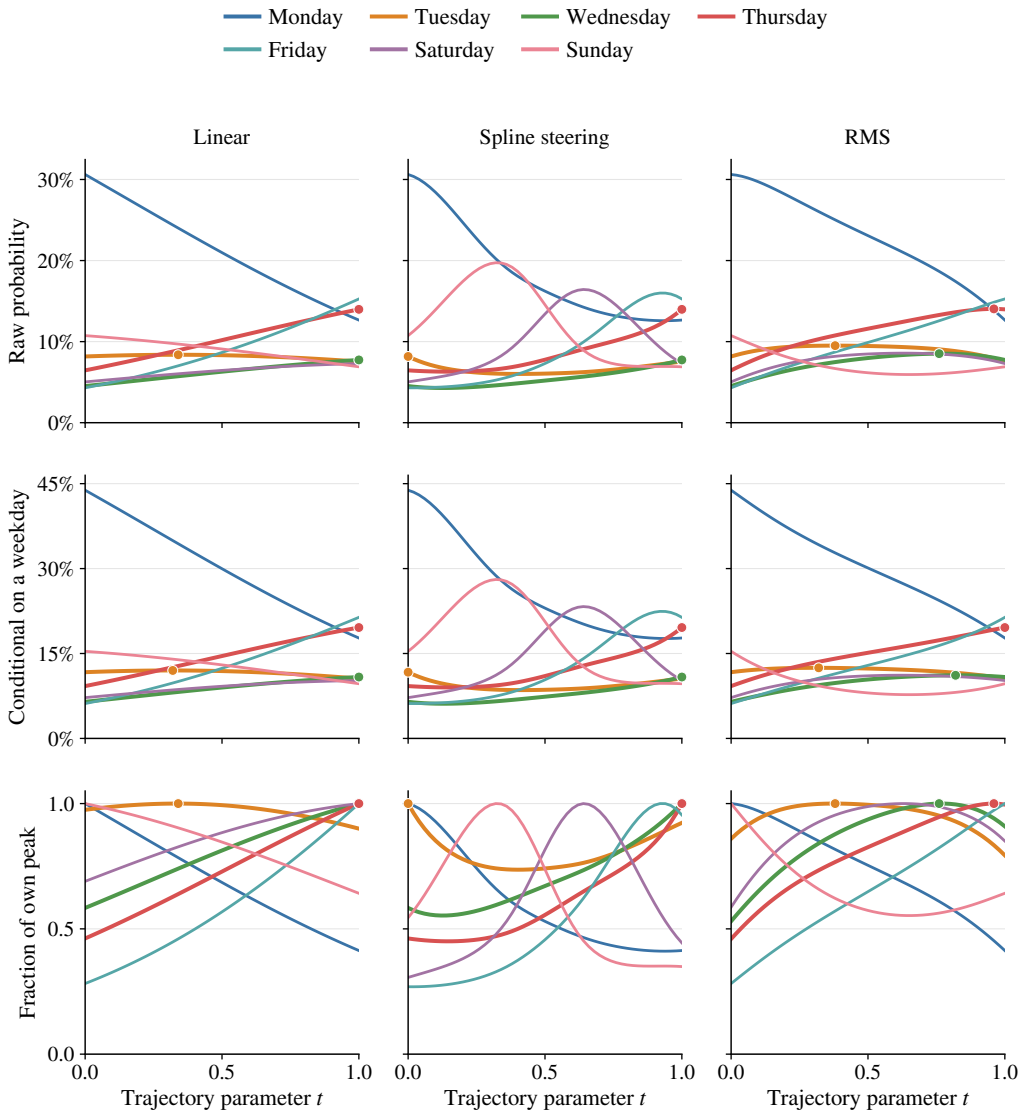

Figure 5: Weekday probabilities along Monday-to-Friday paths. Columns compare the three steering methods. The top row shows probabilities, the middle row shows each day's share of total weekday probability, and the bottom row scales each curve to its own peak to make its shape visible. Dots mark the peaks for Tuesday, Wednesday and Thursday. A peak need not make that day the most probable output.

Dual-space trajectories. Figure 6 displays the same activation-only path and baselines in frozen reference-PCA coordinates. These views show path shape, not full-space support distance. The spline's clearer weekday peaks coexist with higher average output distance. Interpolating centroids does not constrain the path between them to observed support; a centroid can itself lie away from curved data support. This is a possible explanation, not a mechanism isolated by these experiments.

The PCA bases for this figure are fitted to the 823 training references and retain 46.0% of activation variance and 76.9% of output variance. The output embedding is $\sqrt{p}/\sqrt{2}$ over all eight bins. The plotted bounds contain no activation references and 14 output references; points outside these bounds are omitted. Bounds are shared across methods and axes are independently scaled within each panel. Saved waypoints are joined without smoothing. Natural output references retain their own prompts, while paths and decoded centroids use carrier 417.

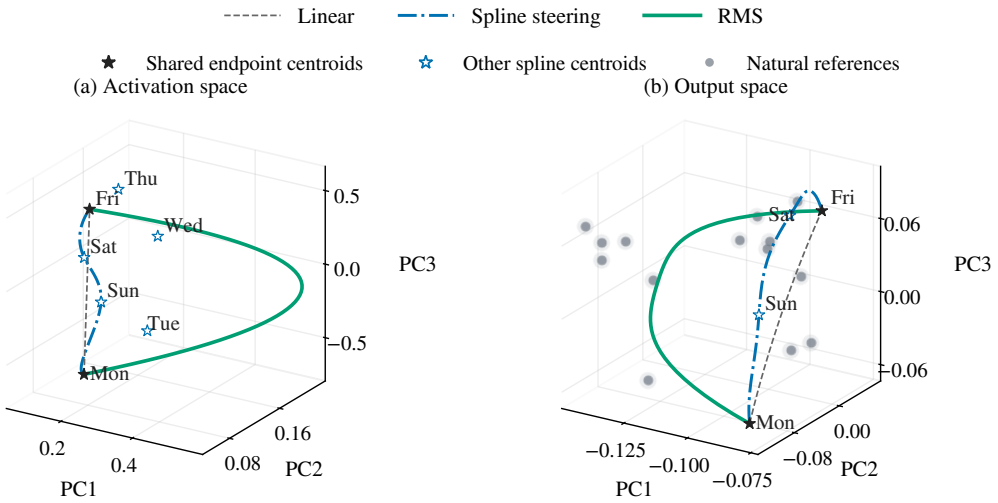

Figure 6: Monday-to-Friday paths in activation and output space. Columns compare linear steering, spline steering and RMS. Stars mark centroids, with filled stars indicating the shared endpoints; grey points show nearby references. These projections illustrate path shape, rather than measuring distance to the reference data.

C.5 SENSITIVITY TO MISSING SPLINE CENTROIDS: AN AGES EXAMPLE

Measuring the effect of an omitted centroid. We remove one intermediate centroid and refit the spline while keeping its endpoints and all other centroid positions fixed. Let γ_{full} be the original spline and γ_{omit} the refitted spline. We compare the same activation and output proximity scores defined in Equations 36 and 37. For either score, write $S[\gamma]$ for its path average and $s(x)$ for the five-neighbor mean at a single activation x inside that average. If t_* is the omitted centroid's position on the original spline, the changes are

$$\Delta_{\text{local}} = s(\gamma_{\text{omit}}(t_*)) - s(\gamma_{\text{full}}(t_*)), \tag{40}$$

$$\Delta_{\text{path}} = S[\gamma_{\text{omit}}] - S[\gamma_{\text{full}}]. \tag{41}$$

Positive values mean that removing the centroid moves the spline farther from observed data; negative values mean it moves closer. The local score checks the position where the centroid used to constrain the path, while the path score checks the average effect. These are distances to observed data, not distances to the omitted centroid. For the age 37-to-42 example, all original centroid positions fall exactly on the saved sampling grid.

Age example.

For age 37-to-42, removing an intermediate centroid moves the refitted spline farther from natural data at that centroid's original trajectory time (Figure 2). We choose the omitted centroid randomly, near the midpoint, or in a dense or sparse region, and refit only the spline. The four rules select three distinct centroids. Local activation distance increases by 0.173–0.215 and output distance by 0.134–0.227. This shows local sensitivity to the specified centroids; whole-trajectory changes are mixed, and this single pair does not establish a density effect.

Removal choices and fixed data. The dense and sparse rules use class-average leave-self-out activation-graph 5-NN distance, rather than class counts. Random removal selects age 41, midpoint and sparse removal select age 39, and dense removal selects age 40. All other knots and the linear and RMS paths stay fixed. Distances use 51 trajectory times and the same five-neighbor evaluations as the main age experiment, including all 92 output bins. References for the omitted age remain available, so this is not a held-out-age evaluation.

C.6 WHAT HAPPENS IF WE ALSO PENALIZE DECODED OUTPUT DISTANCE

Activation support is sufficient for the aggregate proximity wins in the main results. We can additionally penalize the decoded path's distance from observed outputs:

$$\min_{\psi:\, \gamma_\psi(0)=x_0,\, \gamma_\psi(1)=x_1} \widehat{L}[\gamma_\psi] + 10\widehat{R}[\gamma_\psi] + \lambda_{\text{out}} D_{\text{out}}(\gamma_\psi). \tag{42}$$

This term directly targets output proximity, but the optimized variables remain the activation bridge. We have not run a separate optimization over free output distributions followed by inversion to activations.

Output-support score. We fit centered, unwhitened PCA to the 823 natural Hellinger vectors $\sqrt{p_j^{\text{obs}}}/\sqrt{2}$. The fixed 95% variance rule retains seven of eight directions (97.746% variance). For basis U and mean μ, let $z(p) = U^\top(\sqrt{p}/\sqrt{2} - \mu)$ and define

$$\bar{d}_5^{\text{PCA}}(x) = \frac{1}{5} \sum_{j \in \mathcal{N}_5^{\text{PCA}}(x)} \|z(p_c(x)) - z(p_j^{\text{obs}})\|_2, \tag{43}$$

$$D_{\text{out}}(\gamma) = \frac{1}{51} \sum_{i=0}^{50} \left[1 - \exp\left(-\bar{d}_5^{\text{PCA}}(\gamma_i)/0.2\right)\right]. \tag{44}$$

Here $\gamma_i = \gamma(i/50)$. Neighbors are selected in this projected space. The score increases with distance, is bounded by one, and has no hinge or off-PCA residual. Evaluation still uses full eight-bin Hellinger distance, not this score. One could add an ideal constraint $p_c(\gamma([0,1])) \subseteq \mathcal{M}_p$ and replace its path indicator by this finite penalty, with the same non-equivalence caveat as for the activation constraint. The observed natural-output cloud need not equal the fixed-context image $p_c(\mathcal{M})$.

Measured trade-off, not a required component. At fixed activation weight 10, adding output weight 10 lowers the 21-pair mean output distance by 11.6%, from 0.08244 to 0.07286, but raises activation distance by 13.7%, from 0.08364 to 0.09506 (Table 6). Output distance improves on 20/21 pairs; activation distance worsens on all 21. Thus the added term does not improve both metrics simultaneously.

Table 6: Optional output-support ablation across the same 21 pairs. Activation weight is fixed at 10 and length weight at one. The added term changes only output weight, from zero to 10. Lower proximity scores are better; $\pm$ describes the spread of pair means as in Table 2, not independent-run uncertainty.

Method	Output Hellinger	Activation pullback
RMS	0.08244 ± 0.00187	$\mathbf{0.08364} \pm 0.00164$
With output support	$\mathbf{0.07286} \pm 0.00164$	0.09506 ± 0.00200

The output-supported variant has 53/105 interior peaks, versus 55/105 for the main method; both have 7/105 dominant interior peaks. Median increase among interior peaks is 1.116 versus 0.722 percentage points. Strictly ordered interior paths occur on 9/21 versus 8/21 pairs, and ordered dominant paths on 1/21 versus 0/21. These differences do not establish reliable ordered dominance. In Monday-to-Thursday, the added term creates an interior Wednesday peak (12.13% at $t = 0.72$) but lowers Tuesday's raw peak from 9.29% to 8.57%; neither day becomes dominant.

Original exploratory weight grid. Table 7 retains the complete Monday-to-Thursday grid. The original grid selected $(10, 10)$ before the activation-only controls were available; the main method now uses the simpler $(0, 10)$ configuration. At all five tested matched activation-weight comparisons, adding output support lowers output distance but raises activation distance. This is a one-pair exploratory result, not a universal trade-off or optimality claim. Equal coefficients do not mean equal objective values or gradients. The $(10, 10)$ path has greater pullback length than linear (0.27757 versus 0.22203); no no-length control was run.

Table 7: Exploratory Monday-to-Thursday weight grid, retained separately from the activation-only main result. Weights are $(\lambda_{\text{out}}, \lambda_{\text{act}})$, with length weight one. Same actual-network map, seed, initialization and ten updates. One pair and prompt per cell; no independent-run standard error.

Method	Output Hellinger	Activation pullback
Linear	0.09350	0.10622
Spline steering	0.09350	0.10325
$(0, 0)$	0.09407	0.10845
$(1, 0)$	0.08586	0.10342
$(0, 1)$	0.09113	0.10160
$(1, 1)$	0.08572	0.10223
$(3, 1)$	0.08096	0.10189
$(0, 3)$	0.09252	0.09685
$(1, 3)$	0.08346	0.09955
$(3, 3)$	0.08108	0.10078
$(0, 10)$	0.08965	**0.08073**
$(10, 10)$	**0.07973**	0.09956

Dataset visualization. Figure 3 shows density-aware views of the training references and weekday centroids. The projection and its checks are described in Appendix C.1.1.

C.7 AGE METHOD AND EVALUATION

We use the same output metric (Appendix C.2.2), neural-bridge objective (Appendix C.2.5) and support graph (Appendix C.2.4), with the dataset and output bins described in Appendix C.1.2. All reference extraction, model forwards and bridge training use float32, eager attention, and no autocast or TF32. Output Hellinger distances use all output bins, without PCA or conditioning on age-token mass. The activation metric retains the fixed floor $\epsilon = 3.1867797952145338 \times 10^{-6}$.

C.7.1 GRAPH AND OPTIMIZATION SETTINGS

We build a union-20-nearest-neighbor graph on the training reference activations defined in Appendix C.1.2, using eight subsegments for each edge. Query attachments use eight candidates and four subsegments during training, and sixteen candidates and eight subsegments during evaluation. Each pair has its own endpoint-constrained bridge with two hidden layers of width 32, seed 0, ten Adam updates, learning rate 10^{-3} and zero weight decay. Both length and support use 51 uniformly spaced values of t. The support coefficient is 10 and the length coefficient is one; there is no output-support penalty. Earlier-token states retain the age prompt at row 417. Natural output references retain their own prompt contexts.

C.7.2 BASELINES AND EVALUATION

The linear baseline uses the activation chord. Spline steering uses an open natural cubic spline through all 91 full-activation centroids at equally spaced age knots, with zero second derivatives at the global first and last knots; it does not use a periodic calendar construction. The centroid data are specified in Appendix C.1.2. PCAs used for the displayed figure are fitted only to the training references, separately in activation space and in the Hellinger embedding $\sqrt{p}/\sqrt{2}$. These visualization fits do not enter training or the reported distances.

For each method and endpoint pair (Appendix C.1.2), we average mean-5-NN distance over the 51 waypoints, including endpoints, then give every pair equal weight in the task mean. The reported SE is the sample SD of these 21 means divided by $\sqrt{21}$. It is descriptive pair spread; pairs share endpoints, prompt and graph.

C.7.3 ILLUSTRATIVE PAIR AND PROJECTIONS

For the illustration, we use the path scores $S^{\mathrm{act}}[\gamma]$ and $S^{\mathrm{out}}[\gamma]$ from Equations 36 and 37. Let ℓ index endpoint pairs and $\gamma_{m,\ell}$ denote the path for method m. We rank each pair by its average relative improvement over the two baselines in both spaces:

$$q_\ell = \frac{1}{4} \sum_{\sigma \in \{\mathrm{out},\mathrm{act}\}} \sum_{b \in \{\mathrm{linear},\mathrm{spline}\}} \left(1 - \frac{S^\sigma[\gamma_{\mathrm{RMS},\ell}]}{S^\sigma[\gamma_{b,\ell}]}\right).$$

The median of all 21 values selects age 37-to-42, with $q_\ell = 0.10696$. This is a post-hoc representative illustration, not a prespecified test pair or a strongest-win selection. Against the spline, its pullback distance is lower at 38/49 interior output waypoints and 26/49 interior activation waypoints, so the figure must not be read as uniform improvement throughout the path.

The complete run contains all three methods for every sampled pair, with saved probabilities, distances, bridges and ten-update histories. Source, data, graph and artifact hashes were checked before aggregation. Fresh full-model/tail replay and saved graph replay checks report zero discrepancy for the tested values; these numerical checks do not certify exact continuous geodesics or optimization convergence.

Figure projections. The activation and output PCA views in Figure 7 retain 15.2% and 18.4% of reference variance, respectively. They display 154 activation and 68 output references inside the plotted bounds. Both bases are fitted to training references, not paths. All saved waypoints are joined without smoothing. Axes are independently scaled within each panel, with bounds shared across methods. Neither reported proximity metric is computed in the projected space.

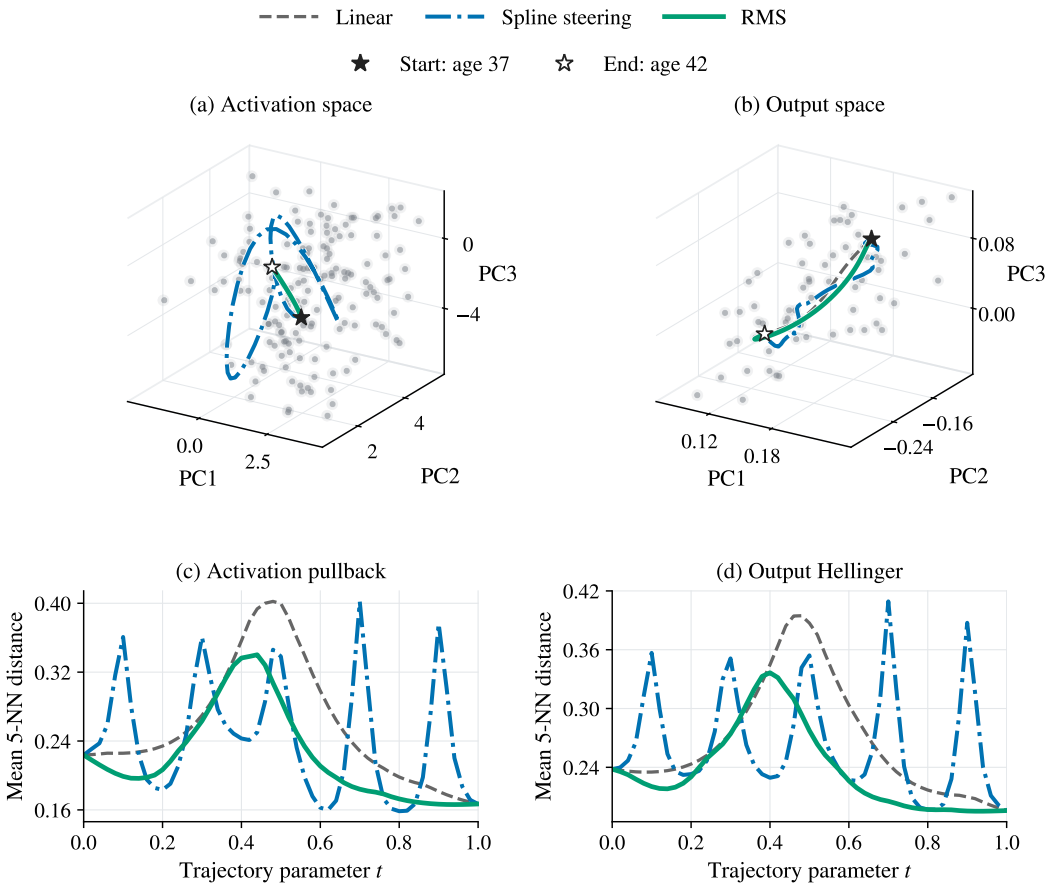

Figure 7: Steering from age 37 to age 42. The top panels show the paths in activation and output space, with reference points in grey and endpoint centroids marked by stars. The bottom panels show distance to the reference data along each path; lower is closer. The projections show path shape, while the distance curves measure proximity in the full spaces.

D SYNTHETIC DATASETS

D.1 DATASET DETAILS

The synthetic tasks separate the geometry of admissible representations from the geometry induced by an output map. Each task supplies a known surface $\mathcal{S} \subset \mathbb{R}^3$, noisy samples embedded in $\mathbb{R}^{24}$, and a prescribed deformation Φ. Given two surface endpoints, we construct the *output-space reference path* as a shortest path on the mapped surface and map it back to activation space, denoting this path by γ_Φ. We compare it with the *activation-space reference path* γ_{round} on the original surface. These reference paths are exact for the sphere and approximated numerically for the other surfaces. This makes the effect of the metric observable separately from whether a path remains near the data.

Samples and embedding. Each dataset contains 5,000 samples. A fixed orthonormal frame $Q \in \mathbb{R}^{24\times 3}$ embeds a surface point u as Qu, and we add independent Gaussian noise with standard deviation 0.01 in each ambient coordinate. Sampling and the embedding use seed 0. The saved encoder split contains 4,513 training and 487 validation examples per surface. Support-score calibration uses its own 10% holdout, specified in Appendix D.2.2. Torus, sphere, and hyperboloid sampling targets uniform surface area; genus-2 uses the projection sampler described below. Figure 8 shows all four sample clouds in their known generating coordinates.

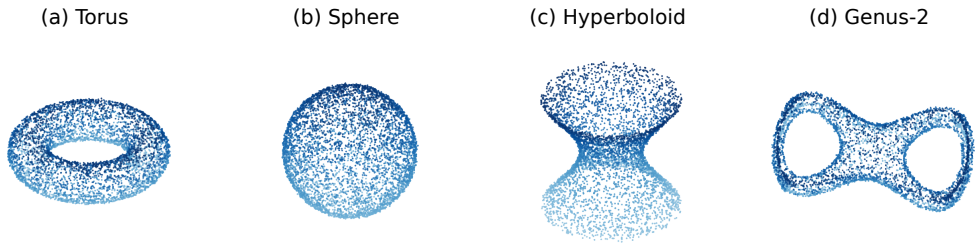

Figure 8: **Synthetic representation datasets.** Each panel shows the sampled points for one surface in its generating coordinates. Color indicates height. The four surfaces provide different geometries for testing steering paths.

D.1.1 SHARED AZIMUTHAL DEFORMATION

We change output-space distances by stretching an angular band while keeping the undeformed sample cloud fixed. In cylindrical coordinates, the three-dimensional map is

$$\Phi_3(\rho\cos\varphi, \rho\sin\varphi, z) = \big(\rho\cos h(\varphi), \rho\sin h(\varphi), z\big). \tag{45}$$

The increasing angular map satisfies $h(0) = 0$, $h(2\pi) = 2\pi$, and

$$h'(\varphi) = c_0\left[1 + (a-1)\exp\{\kappa(\cos(\varphi-\varphi_c)-1)\}\right], \qquad \kappa = \frac{\log 2}{1-\cos(w/2)}. \tag{46}$$

Here $w = 60^\circ$ is the bump's full width at half maximum, $a = 3$ is its nominal stretch, $\varphi_c = 0$ is its center, and c_0 normalizes one full turn. Thus a is a deformation parameter, separate from any support-penalty weight. The ambient map $\Phi(x) = Q\Phi_3(Q^\top x) + (I - QQ^\top)x$ preserves the component orthogonal to the generating frame. The encoder learns pairwise image distances $\|\Phi(x_i) - \Phi(x_j)\|_2$; the support estimator receives the undeformed samples. Figure 9 illustrates the map on the torus.

D.1.2 TORUS: CHOOSING A PATH AROUND A HOLE

The torus has major radius $R = 1$ and minor radius $r = 1/3$, with coordinates $((R + r\cos\psi)\cos\varphi, (R + r\cos\psi)\sin\varphi, r\sin\psi)$. The endpoint pairs straddle the stretched band. The activation-space reference paths and output-space reference paths pass around different sides of the central hole, testing whether the learned metric changes path selection while the support penalty keeps the path near the torus. Reference paths are computed numerically in the known surface chart.

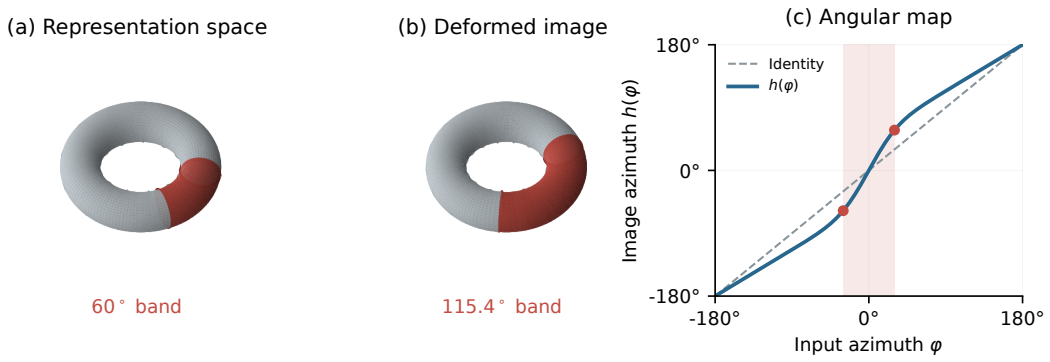

Figure 9: **The deformation changes distances along the surface.** Left: the red band marks a region of the original torus. Middle: the same points spread farther apart after mapping, while the torus retains its overall shape. Right: the angular map shows how positions change relative to the identity map. The deformation can therefore change which path between two points is shortest.

D.1.3 Sphere: choosing a side without a topological hole

The unit sphere satisfies $\|u\|_2 = 1$. Endpoints lie on the equator on opposite sides of the stretched band, separated by 160° before mapping. The output-space reference path follows the other side of the equator from the activation-space reference path. This is a change in the preferred path on a simply connected surface, not a change in homotopy class. The sphere provides a closed-form reference path from its mapped endpoints.

D.1.4 Hyperboloid: choosing a path around a narrow waist

The truncated one-sheet hyperboloid satisfies $u_1^2 + u_2^2 - u_3^2 = 1/4$, with $|u_3| \leq 1$. Endpoints lie on its waist circle, of radius $1/2$, and straddle the stretched band. This task tests the same path change on a surface with negative Gaussian curvature, where an ambient chord can cut through the waist. Reference paths use the known surface metric and numerical search over winding branches.

D.1.5 Genus-2: path selection on an implicit surface

Before multiplication by 1.6, the two-hole surface is the level set

$$\left(u_1^2(1-u_1^2) - u_2^2\right)^2 + u_3^2 = 0.01. \tag{47}$$

The sampler draws points in a bounding box and projects them onto this level set; it does not sample uniformly in surface area. The deformation changes the surface's image, and reference paths are shortest paths on 8-nearest-neighbor graphs of the clean samples before and after mapping. This task uses a preselected cohort with a decisive cost margin between the reference paths. Its path label is defined by proximity to the two graph reference paths, so the conclusion is limited by graph resolution and pair selection (Appendix D.4.1 and Appendix D.4.2).

Section D.1 defines the four surfaces, sampling procedures, and steering tasks. The shared output map is defined in Section D.1.1 and illustrated in Figure 9. Here we specify the estimators, solvers, reference path construction, and evaluation used for these tasks.

D.2 Metric and support estimation

D.2.1 Learning the pullback map

For each synthetic task, the output deformation $\Phi : \mathbb{R}^{24} \to \mathbb{R}^{24}$ is known, but the path optimizer uses $F = f_\theta$ in the notation of the main text, where $f_\theta : \mathbb{R}^{24} \to \mathbb{R}^3$ is a learned encoder with hidden widths $256/128/64$. Its supervised distance target is $d_{ij} = \|\Phi(x_i^{\text{obs}}) - \Phi(x_j^{\text{obs}})\|_2$. We minimize $50\mathcal{L}_{\text{dist}} + \mathcal{L}_{\text{recon}}$, where $\mathcal{L}_{\text{dist}}$ matches pairwise latent distances to d_{ij} and $\mathcal{L}_{\text{recon}}$ reconstructs x_i^{obs}

through the decoder. Thus the optimized pullback is $J_{f_\theta}^\top J_{f_\theta}$; the exact $J_\Phi^\top J_\Phi$ is used only in named oracle controls.

D.2.2 SUPPORT-PENALTY CALIBRATION

We use the pointwise penalty $r(x)$ defined in Appendix B.4.1, Equation 27, with no generated negatives. The reference set contains 90% of the observed activations. We set τ to 1.5 times the median ten-nearest-neighbor distance of the held-out activations to this reference set, measured in principal-component coordinates.

D.3 PATH OPTIMIZATION

D.3.1 ENERGY WITH A SEPARATE SUPPORT PENALTY

The synthetic experiments use energy in place of the length objective in Equation 4. With the learned pullback metric $g = J_{f_\theta}^\top J_{f_\theta} + \epsilon I$, we define

$$E[\gamma] = \int_0^1 \dot\gamma^\top g(\gamma)\dot\gamma \, dt = \int_0^1 \left(\|J_{f_\theta}(\gamma)\dot\gamma\|_2^2 + \epsilon \|\dot\gamma\|_2^2 \right) dt, \tag{48}$$

and minimize $E[\gamma] + \lambda_E R[\gamma]$. Without the support penalty, energy minimization gives shortest paths traversed at constant metric speed: $E[\gamma] \geq L[\gamma]^2$, with equality at constant speed. Adding $R[\gamma]$ can make the two formulations have different minimizers, so their penalty weights are chosen separately. The LLM experiments retain the length formulation (Appendix C.2.2).

RMS penalizes the support score separately, without an additional square. The discrete objective and its scaling are specified below; the matched torus comparison in Appendix D.6.1 motivates the choice of energy for the synthetic suite.

D.3.2 WAYPOINT ENERGY OPTIMIZATION

Let $\gamma_0, \ldots, \gamma_N$ define $N = 64$ segments with fixed endpoints $\gamma_0 = x_0$ and $\gamma_N = x_1$, $\Delta_i = \gamma_{i+1} - \gamma_i$, and $\bar\gamma_i = (\gamma_i + \gamma_{i+1})/2$. We minimize

$$E_{\text{code}}(\gamma_{0:N}) = \sum_{i=0}^{N-1} \left(\|J_{f_\theta}(\bar\gamma_i)\Delta_i\|_2^2 + 10^{-3}\|\Delta_i\|_2^2 \right), \tag{49}$$

$$J_E(\gamma_{0:N}) = E_{\text{code}}(\gamma_{0:N}) + \lambda_E \frac{1}{N+1} \sum_{i=0}^{N} r(\gamma_i). \tag{50}$$

The support score is not squared and is not differentiated as an additional coordinate of the output map. The $10^{-3}I$ metric floor remains within energy. The continuous-energy convention is $\bar E_N = N E_{\text{code}}$; our coefficients apply to the unscaled E_{code}. Rescaling only energy while keeping λ_E fixed would change the relative penalty.

D.3.3 SOLVER SETTINGS AND INITIAL PATHS

We use float64 L-BFGS with strong-Wolfe search, history 20, at most 30 outer and 100 inner iterations, and a gradient-infinity-norm target of 10^{-6}. Forward-mode Jacobian–vector products retain the waypoint dependence of the metric. We select the lowest total objective over the initial paths. Torus uses the chord, a support-graph path, and the alternative torus arc; sphere and hyperboloid additionally include both ring arcs. Genus-2 uses the chord, a preimage-graph path, and an alternative graph path obtained by excluding nodes near the middle of the first path. These initializations use known geometry or sampled support; they are not evidence for discovery without geometric information.

D.4 REFERENCE PATHS AND EVALUATION

D.4.1 REFERENCE PATH CONSTRUCTION

We construct the output-space reference path γ_Φ (mapped back to activation space) and the activation-space reference path γ_{round} from the known geometry, as defined in Section 4. For the

torus (Section D.1.2) and hyperboloid (Section D.1.4), we numerically minimize chart energy under the exact first fundamental form over the implemented winding branches. The sphere reference path is the closed-form great circle (Section D.1.3). For genus-2 (Section D.1.5), we build separate 8-nearest-neighbor graphs in the undeformed and deformed coordinates of the same clean sample. Graph connectivity and edge lengths are determined in the respective coordinates. We map the image-graph path back using its sample indices. Thus the genus-2 reference path is limited by graph resolution; it is not an analytic geodesic or a certificate of homotopy class.

D.4.2 Endpoint selection

The band-straddling torus, sphere, and hyperboloid tasks give a binary path signature under the deformation in Section D.1.1. We screen every endpoint pair before training and retain it only when the two reference paths are separated by more than three support-score resolution units. For genus-2, we additionally require an image-cost margin above 8%; only the resulting decisive pairs support a path claim.

D.4.3 Baselines and penalty-weight selection

RMS uses Equation 50. Pullback only sets $\lambda_E = 0$. Activation support only replaces $f_\theta(x)$ by the centered PCA coordinates $z(x)$ while retaining the support penalty and ambient metric floor. It therefore uses Euclidean energy in the PCA span without a learned pullback metric. This baseline inherits RMS's selected penalty weight without retuning. Endpoints, samples, initializations, and solver settings are matched across the three methods; RMS and pullback only use the same frozen encoder.

For each geometry, pairs 0 and 4 select λ_E from $\{0.01, 0.1, 1, 10, 100\}$. We rank candidates by correct-path count, then worst maximum waypoint-to-surface distance, then mean path deviation, and finally smaller weight. We freeze the selected weight before evaluating pairs 1, 2, 3, 5, 6, and 7. The selected weights are 100 for torus, sphere, and hyperboloid, and 10 for genus-2. This is an exploratory comparison on previously inspected endpoint cohorts, including the frozen decisive genus-2 pairs, rather than a fresh held-out benchmark.

D.5 Evaluation metrics

We evaluate the optimized activation path γ against the known surface and two reference paths: γ_Φ, the output-space reference path mapped back to activation space, and γ_{round}, the activation-space reference path. Their construction is given in Appendix D.4.1.

D.5.1 Path recovery

Path recovery checks whether our path goes the same way around the surface as the output-space reference path, for example, around the same side of the torus hole. For the torus, sphere and hyperboloid, $\mathrm{dir}(\gamma)$ records the path's direction around the vertical axis. For the two-hole surface, we instead compare its deviation from the two reference paths:

$$\mathrm{Recovery}(\gamma) = \begin{cases} \mathbf{1}\{\mathrm{dir}(\gamma) = \mathrm{dir}(\gamma_\Phi)\}, & \text{torus, sphere, hyperboloid,} \\ \mathbf{1}\{\mathrm{Dev}(\gamma, \gamma_\Phi) < \mathrm{Dev}(\gamma, \gamma_{\mathrm{round}})\}, & \text{two-hole surface.} \end{cases} \tag{51}$$

Here $\mathbf{1}$ is one when its condition holds and zero otherwise. Dev is defined below. Recovery checks the path choice; it does not require an exact match to the reference path or certify a homotopy class. The numerical direction checks are specified in Appendix D.5.4.

D.5.2 Path deviation

Path deviation measures how closely our path follows the output-space reference path. We compare points at the same fraction of the distance along each path, for example, their halfway points, and average the gaps between them. For any two paths γ and η, we sample K points evenly along each path's Euclidean arc length. Let $\bar{\gamma}_j$ and $\bar{\eta}_j$ be the points at fraction $j/(K-1)$ of their respective

path lengths. We compute

$$\mathrm{Dev}(\gamma, \eta) = \frac{1}{K} \sum_{j=0}^{K-1} \|\bar{\gamma}_j - \bar{\eta}_j\|_2. \tag{52}$$

We report $\mathrm{Dev}(\gamma, \gamma_\Phi)$, using Euclidean distances in activation space. This score is not a nearest-neighbor distance between paths and is not optimized directly.

D.5.3 Surface drift

Surface drift measures how far our path leaves the surface at its worst point. We check points along the path, including between optimized waypoints, and report the largest estimated distance to the known, noise-free surface $\mathcal{M}$. Let $\widehat{d}_{\mathcal{M}}(x)$ estimate the Euclidean distance from x to that surface. We report

$$\mathrm{Drift}(\gamma) = \max_j \widehat{d}_{\mathcal{M}}(\gamma(t_j)). \tag{53}$$

The times t_j sample both the optimized waypoints and the straight segments between them. Thus a segment that cuts through the surface's interior can increase drift even when its endpoints lie on the surface. The point-to-surface distance is exact for the torus and sphere, numerical for the hyperboloid, and a first-order approximation for the two-hole surface. This sampled maximum does not bound every point of the continuous path. The distance calculations are described below; none uses the learned support penalty $r(x)$.

D.5.4 Numerical evaluation details

Direction checks. We project the saved waypoints to their known generating coordinates, $y_i = Q^\top \gamma_i$, and compute their angles around the vertical axis. We unwrap these angles to remove jumps of 2π, then subtract the first angle from the last. For the selected torus endpoints, the output-space reference path takes the -200° direction rather than the original $+160^\circ$ direction; recovery requires angular travel within 1° of -200°. For the sphere and hyperboloid, recovery requires negative total angular travel, matching their output-space reference paths. For the two-hole surface, the strict comparison in Equation 51 treats a tie as not recovered.

Sampling and aggregation. We use $K = 200$ points per path for deviation. For drift, we evaluate $t_j = j/(J-1)$ with $J = 1{,}025$ on the saved piecewise-linear path. This samples all 64 segments at the same resolution and includes all 65 waypoints. Across the six evaluation pairs, we report the count of recovered paths and the medians of path deviation and sampled maximum drift.

Distance to each surface. The known orthonormal frame Q separates an activation into its three-dimensional coordinates $y = Q^\top x$ and its orthogonal residual $x - Qy$. We combine distance within these coordinates and the residual norm by the Pythagorean theorem. This generating frame is distinct from the estimated PCA basis used by the support penalty. For the sphere, the within-span distance is $|\|y\|_2 - 1|$. For the torus, it is the absolute difference between the distance to the major-radius circle and the minor radius. Both give exact ambient distances. For the truncated hyperboloid, we approximate the closest point by searching 401 equally spaced heights in $[-1, 1]$, at the query point's angle. For the two-hole surface, we use the first-order estimate $|f_{\mathcal{S}}(y)|/\max(\|\widetilde{\nabla} f_{\mathcal{S}}(y)\|_2, 10^{-9})$, where $f_{\mathcal{S}}$ is its scaled implicit surface function (Appendix D.1.5). Specifically, with $u = y/1.6$, $f_{\mathcal{S}}(y) = (u_1^2(1-u_1^2) - u_2^2)^2 + u_3^2 - 0.01$. We approximate its gradient with respect to y by centered finite differences with step 10^{-6}. This local distance estimate can be inaccurate far from the surface.

D.6 Synthetic results and numerical checks

RMS meets the path recovery criterion for all six pairs on each surface and has lower median surface drift than pullback only (Table 1). Activation support only meets this criterion for no pairs, despite having slightly lower drift than RMS on the sphere and hyperboloid. RMS has lower path deviation than both baselines except on genus-2, where pullback only has lower deviation (0.04413 versus 0.05348).

D.6.1 ENERGY VERSUS LENGTH AND NUMERICAL LIMITS

In the matched torus comparison, RMS with energy recovers $6/6$ evaluation paths versus $5/6$ for midpoint length plus the same unsquared score, with independently selected coefficients. Median maximum surface drift is 0.00158 versus 0.00387. This motivates energy as the primary objective for the broader rerun; it does not establish its superiority on every geometry. None of the selected paths meets the requested 10^{-6} gradient tolerance. These results establish finite-resolution path and support outcomes, not stationary solutions or globally shortest paths.

D.7 SYNTHETIC PATH VISUALIZATIONS

No selected activation support only path reaches the stationarity target.

The following panels use evaluation pair 1 for each surface, fixed before inspecting the new control outcomes. They show actual solver paths without projection back onto the surface. Display coordinates are the known three-dimensional span and its image under the known deformation; distances in the table are evaluated in the full ambient space. The reference paths are evaluation targets, not exact recovered paths.

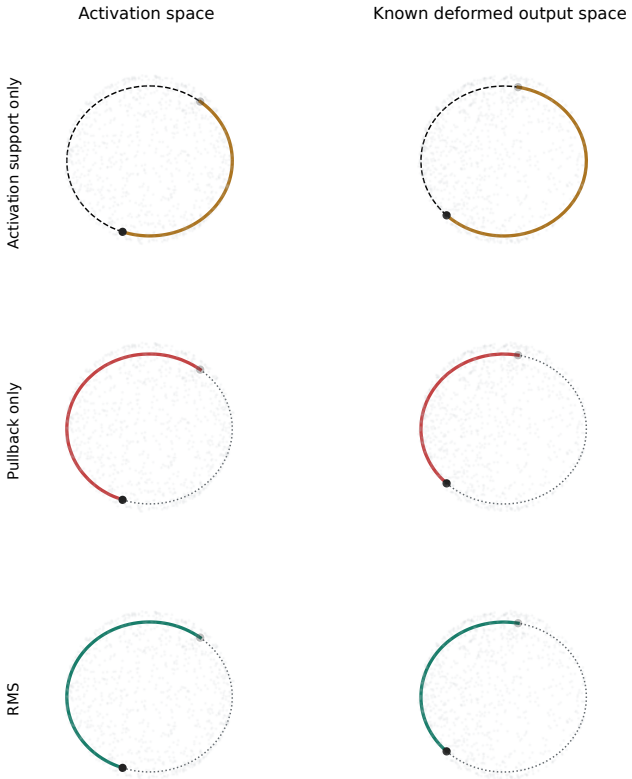

Figure 10: Sphere steering paths. Rows compare activation support only, pullback only, and RMS. The left column shows activation space and the right shows the known deformed output space. Dashed curves show output-space reference paths; dotted curves show activation-space reference paths. Both are shown in each panel's display space.

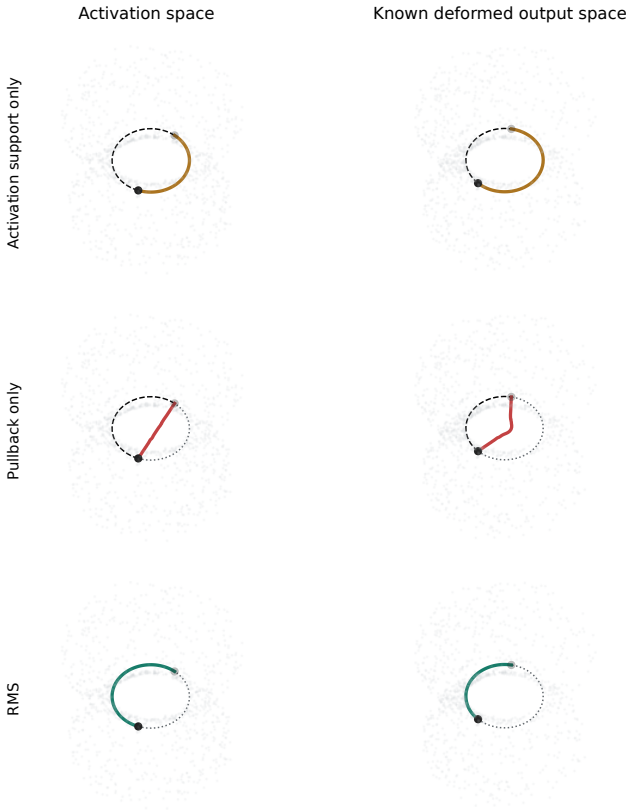

Figure 11: Hyperboloid steering paths. Rows compare activation support only, pullback only, and RMS. The left column shows activation space and the right shows the known deformed output space. Dashed curves show output-space reference paths; dotted curves show activation-space reference paths. Both are shown in each panel's display space.

Visualization details. Figure 8 displays all 5,000 noisy samples per surface, projected from 24 dimensions into the known generating frame, with equal spatial units within each panel. Figure 1 uses torus evaluation pair 1, fixed before inspecting the new control outcomes. Its output panels use the known deformation rather than the learned encoder. Paths are overlaid for visibility without projecting them onto the surface; they are numerical approximations rather than certified geodesics.

E STEERING IN THE WILD: EXPERIMENTAL DETAILS

These details apply to the hallucination experiment in Section 5.3. For a prompt q, let x_q be its natural activation and $\gamma(1)$ the steered endpoint. We write $p_q(x)$ for the full next-token distribution after replacing the activation at the intervention site by x, with the prompt and model fixed.

E.1 PROMPTS AND ACTIVATION MEANS

We use the original hallucination training pairs and open-ended questions of Rimsky et al. (2024). For the CAA direction v, we average the layer-15 post-block activations separately over the behavior-matching and contrasting answers from all 1,000 training pairs, then subtract the latter mean from the former. These activations are extracted at the original answer-letter position. We use the raw difference without normalizing it. The means define a direction; neither mean is used as a path endpoint.

The calibration set contains ten of the 50 official open-ended questions, selected with seed 0 before inspecting model outputs. The remaining 40 questions were reserved. Each question is presented as `[INST] question [/INST]`, without an A/B answer prefix. Its target and contrasting answer texts come from the aligned paired-answer version of the dataset, with the A/B markers removed. The target is the hallucination-aligned answer in this experiment.

We use the frozen Llama-2-7b-chat model and intervene at the layer-15 activation of the final prompt token. This is a different token position from the answer-letter activations used to construct v. One bank of 16 training questions supplies natural activations at this open-answer prompt position. The bank is fixed across methods and penalty weights; we decode it in each query prompt's context when constructing the support graph and evaluating output proximity. The intervention replaces the activation at the original prompt position only; it is not repeatedly applied at newly generated tokens.

E.2 NEURAL BRIDGE WITH A FREE ENDPOINT

The bridge in Appendix C.2.3 fixes both endpoints. Here we fix only the source x_q and make the terminal activation x_1 a trainable parameter:

$$\gamma_{\psi,x_1}(t) = x_q + t(x_1 - x_q) + 4t(1-t)B_\psi(t), \qquad t \in [0,1]. \tag{54}$$

The correction still vanishes at both ends, so $\gamma(0) = x_q$ and $\gamma(1) = x_1$ for every choice of parameters. Unlike the original bridge, B_ψ takes only t as input, without endpoint-embedding networks. Each prompt has its own bridge. The correction network has widths $1 \to 32 \to 32 \to 4096$, with Tanh activations after the first two affine layers. Its final layer is initialized to zero, and x_1 is initialized at $x_q + v$, so the initial path is exactly the mean-difference line. We optimize x_1 and ψ jointly while the language model stays frozen.

The parameter t indexes candidate activations for the same fixed prompt. Traversing the path does not correspond to generating successive tokens; the endpoint supplies the intervention used for the reported answer scores.

E.3 BASELINES AND NUMERICAL SETTINGS

Equation 6 retains $L[\gamma]$ and $R[\gamma]$ from Equation 4. The metric uses the full first-next-token Hellinger map with the activation-space floor $\epsilon = 7.416704174829647 \times 10^{-6}$. We approximate length by the polygonal sum in Equation 32, and approximate $R[\gamma]$ by the mean pointwise support penalty at 21 equally spaced path parameters, including the endpoints. The penalty is the mean distance to the five nearest training activations under the pullback-weighted graph, as in Appendix B.4.2. Each

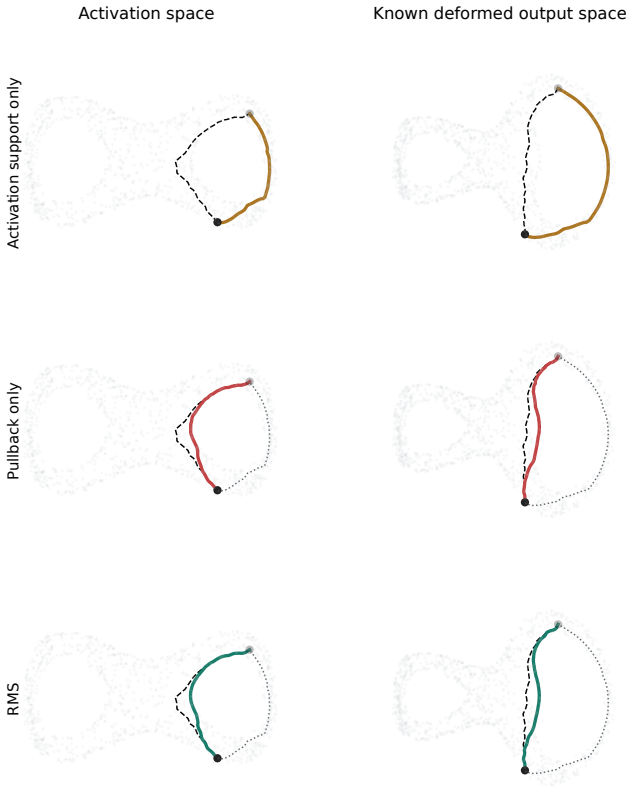

Figure 12: Genus-2 steering paths. Rows compare activation support only, pullback only, and RMS. The left column shows activation space and the right shows the known deformed output space. Dashed curves show output-space reference paths; dotted curves show activation-space reference paths. Both are shown in each panel's display space.

prompt's graph uses eight neighbors, four subdivisions per edge and query connection, and eight candidate query connections.

RMS uses Equation 54. The optimized-straight control sets $B_\psi = 0$ and optimizes its own endpoint under the same objective. Both families use $\beta = 1$, seed 0, 30 Adam updates with learning rate 0.001, and $\lambda \in \{0, 1, 10\}$. We retain the lowest-objective candidate encountered, including a no-movement candidate. Pullback length remains in the objective when $\lambda = 0$.

Tuned mean-difference steering restricts the path to $\gamma_\alpha(t) = x_q + t\alpha v$ and fixes $\lambda = 0$. For each prompt, we optimize $\alpha \in [0, 8]$ using 17 initial candidates and two rounds of midpoint refinement around the three best candidates, retaining the lowest objective. This single baseline is compared with every support weight. The scalar search and Adam runs have different optimization budgets. We also report the natural start and the untuned $\alpha = 1$ endpoint. All optimized methods use the same paired-answer steering score; this is not a comparison at fixed achieved steering strength.

E.4 Steering strength

Each prompt has a target answer y_+ and a contrasting answer y_-. We score each answer by the mean log probability of its tokens, conditioning each token on the preceding answer tokens (teacher forcing). Steering strength is the difference between these two scores:

$$b_q(x) = \frac{\log P(y_+ \mid q; x)}{|y_+|} - \frac{\log P(y_- \mid q; x)}{|y_-|}. \tag{55}$$

Here $|y_+|$ and $|y_-|$ count answer tokens, excluding the end-of-sequence token. We use natural logarithms. Larger values favor the target answer. In this experiment, y_+ is aligned with hallucination, so this score measures preference for that behavior rather than factual accuracy.

Table 8 reports $b_q(\gamma(1))$. The optimization reward instead uses the gain $b_q(\gamma(1)) - b_q(x_q)$ in Equation 6. Thus the reported steering strength measures the achieved endpoint preference, using the same answer score that guides endpoint optimization.

E.5 Output Hellinger distance

For each prompt, we decode the observed activations x_j^{obs} under that same prompt, obtaining outputs $p_q(x_j^{\mathrm{obs}})$. We compare the endpoint output with the five closest of these distributions. Their mean Hellinger distance is

$$\overline{H}_q(\gamma(1)) = \frac{1}{5} \sum_{j \in \mathcal{N}_5^{\mathrm{out}}(\gamma(1))} H\big(p_q(\gamma(1)), p_q(x_j^{\mathrm{obs}})\big), \tag{56}$$

where $H(p, p') = \|\sqrt{p} - \sqrt{p'}\|_2 / \sqrt{2}$ and $\mathcal{N}_5^{\mathrm{out}}(\gamma(1))$ contains the indices of the five nearest decoded outputs under this distance. We use the full next-token vocabulary without PCA. Lower values mean that the endpoint output is closer to outputs obtained from observed activations.

The neighbors are selected in output space; graph distances do not enter this score. The same training-activation bank described in Appendix E.1 also supplies the activation-support penalty. This direct output distance is an evaluation measure and is not added to the training objective. Unlike the Toy LLM path averages, this score is evaluated only at the endpoint.

E.6 Aggregation and evaluation scope

For each method and penalty weight, we average each endpoint score equally over the ten calibration prompts. Table 8 reports these descriptive means from one optimization seed. The prompts and paired answers used to optimize endpoints also supply the steering strength evaluation, and the support bank is shared with the output-proximity evaluation. These comparisons do not constitute an independent held-out test. The methods optimize their own endpoints, so the comparison does not hold steering strength fixed.

Method	λ	Hallucination		Corrigibility		Myopia	
		Steering strength ↑	Mean Hellinger ↓	Steering strength ↑	Mean Hellinger ↓	Steering strength ↑	Mean Hellinger ↓
Unsteered baseline	—	0.5528	0.0434	-0.8716	0.0062	-3.9184	0.0312
Mean difference, $\alpha = 1$	—	0.5569	0.0222	-0.2923	0.0094	-3.8954	0.0204
Tuned mean difference	0	0.6262	0.0265	4.4879	0.1522	-1.7721	0.1294
Optimized straight	0	0.7529	0.0418	3.0994	0.0105	-1.3556	0.0568
Optimized straight	1	0.7575	0.0375	3.0989	0.0101	-1.3581	0.0528
Optimized straight	10	0.7334	0.0098	3.0836	0.0082	-1.4802	0.0245
RMS	0	0.7564	0.0367	3.0980	0.0107	-1.3609	0.0603
RMS	1	0.7604	0.0345	3.0976	0.0103	-1.3617	0.0538
RMS	10	0.7487	0.0086	3.0874	0.0084	-1.4328	0.0294

Table 8: Steering strength and output proximity on three behaviors. Ten calibration prompts per behavior. Higher steering strength means greater preference for the behavior-aligned answer; lower Hellinger distance means closer outputs to those obtained from observed activations. The hallucination columns correspond to the comparisons in the text. All entries average the full ten-prompt cohort.

E.7 Detailed results

The support penalty trades steering strength for output proximity. Across hallucination, corrigibility, and myopia, increasing the activation-support penalty from $\lambda = 0$ to $\lambda = 10$ lowers mean output Hellinger distance for both optimized path families, while reducing steering strength (Table 8). The lower Hellinger distances indicate closer proximity to outputs decoded from observed activations, as defined in Section 4.

Curvature helps on hallucination, but not consistently across tasks. Curvature provides additional gains on hallucination: at $\lambda = 10$, RMS achieves a higher steering score than optimized straight paths (0.7487 versus 0.7334) and lower Hellinger distance (0.0086 versus 0.0098). However, this advantage is not consistent across tasks. On corrigibility at $\lambda = 1$, RMS has a small steering-score difference of -0.00127 relative to optimized straight paths (95% CI $[-0.00260, -0.00015]$) and higher Hellinger distance, with a difference of $+0.000199$ (95% CI $[+0.000044, +0.000407]$). On myopia at $\lambda = 10$, curvature increases steering strength but also increases reference distance, exposing a trade-off between the two measures.

Tuned mean-difference steering reaches the strongest corrigibility score. Tuned mean-difference steering additionally achieves the strongest corrigibility score, although with the largest reference distance.

Summary and hypotheses Together, these results support using the activation-support penalty to control reference proximity and show that curvature can improve steering, while identifying settings where straight paths perform better. We hypothesize that some behaviors might be approximately linearly represented which could explain the limited gains from curvature. As an alternative hypothesis, steering between distinct activation clusters may require crossing sparsely supported regions, which could explain why stronger support penalties restrict steering strength. We will further explore the limits of this data-dependent approach of steering in depth in our future work and address this in Section 6. We optimize and evaluate endpoints using the same paired answers; Appendix E.6 describes the evaluation scope.