STEERING LANGUAGE MODELS WITHOUT CONTRASTIVE PAIRS VIA UNSUPERVISED LATENT REGION MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Activation steering methods control language model behavior by intervening on internal activations, but many steering-vector approaches require contrastive prompts or paired good and bad completions. We study a lower-supervision setting where only a behavior dataset and the model's own zero-shot choices are available. We introduce Unsupervised Latent Region Matching (ULRM), which splits examples into successful and unsuccessful subsets and learns a nonlinear map from unsuccessful prompt-side decision activations toward the successful activation region. Our loss objective encourages the intervention to shift activations toward preferred behavior while preserving task-relevant information.

We evaluate ULRM on two sycophancy datasets using Llama-3.1-8B-Instruct and Qwen2.5-7B-Instruct. We compare two vector baselines. MeanDiff subtracts the mean unsuccessful activation from the mean successful activation in the same split. Contrastive Activation Addition (CAA) Panickssery et al. (2024) averages paired preferred-minus-nonpreferred completion activations. Despite CAA's stronger paired supervision, our best-performing method, MMD-ULRM, leads both baselines in all eight comparisons, averaging 0.599/0.528 accuracy/probability versus MeanDiff's 0.527/0.487 and CAA's 0.503/0.457. In a matched-update data-scaling study, MMD-ULRM is strongest on average from 624 training examples onward. Full-data experiments further show that performance can depend on training-set composition, especially for CAA. These results suggest that behavior-split latent-region matching is a promising route to practical activation steering without paired contrastive completions.

1 INTRODUCTION

Activation steering offers a direct way to study and control the behavior of large language models. Rather than changing model weights or prompts, steering methods intervene on hidden states. This makes them especially relevant for mechanistic interpretability: a successful intervention is both a behavioral tool and evidence about where behavior-relevant information is represented.

The most natural supervision for steering is often not available. Activation additions, inference-time interventions, and related representation-engineering methods commonly define an intervention by estimating a direction or subspace associated with a concept or behavior Turner et al. (2024); Li et al. (2024); Zou et al. (2025). Contrastive Activation Addition (CAA) Panickssery et al. (2024) does this from paired good and bad completions for the same prompt. Such pairs are clean and powerful, but they are also a strong data requirement. In many settings we may only have a dataset on which the model sometimes behaves as desired and sometimes does not.

We propose to use this lightweight signal directly. We run the model on a behavior dataset, split examples by whether the model exhibits the preferred behavior, and collect layer activations from the resulting two groups. We then view these activations as samples from two latent regions: a preferred-behavior region and a nonpreferred-behavior region. Steering becomes a problem of moving an activation from one region toward the other while preserving the prompt information that should not change.

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.

This changes the object being learned. A steering vector estimates a single displacement, direction, or low-dimensional subspace in activation space. ULRM instead learns a map between two empirical activation clouds. This is useful when the difference between successful and unsuccessful behavior is not well described by one global translation, or when examples cannot be paired cleanly at the completion level.

Our experiments support three claims. First, MMD-ULRM outperforms the unsteered model, MeanDiff, and CAA across both models and data pools despite using no paired completions. Second, its advantage depends on nonlinear mapping and identity preservation. Third, when we vary the number and class balance of training examples, MMD-ULRM is strongest on average once the balanced set reaches 624 examples, while the other methods are more sensitive to data size or composition.

2 Related Work

Training-time alignment. Training-time alignment methods such as Ouyang et al. (2022) and Bai et al. (2022) improve model behavior by updating model parameters using demonstrations, output rankings, preference comparisons, or reward-model signals. These methods are powerful, but their data requirements are heavier in two ways: they require stronger annotations and substantially larger numbers of labeled samples than the behavior-split setting we study. ULRM is complementary. It keeps the model frozen and asks what can be done from the much smaller signal of which examples the model already answers in the preferred or nonpreferred way.

Activation and representation steering. A growing line of work controls language models by editing hidden states rather than weights or prompts. Examples include latent steering vectors, activation additions, inference-time interventions, representation engineering, contrastive activation addition, conceptor-based steering, preference-trained representation steering, and hypernetwork-generated steering vectors Subramani et al. (2022); Turner et al. (2024); Li et al. (2024); Zou et al. (2025); Panickssery et al. (2024); Postmus & Abreu (2025); Wu et al. (2025); Sun et al. (2025). These methods differ in architecture and objective, but they typically rely on a higher-supervision interface: contrastive prompts or completions, labeled examples, curated concepts, preference data, or large collections of steering tasks. ULRM shares their inference-time intervention perspective, but asks how much steering signal can be extracted when the only supervision is the model's own observed preferred/nonpreferred behavior split.

Painless Activation Steering (PAS) also partitions examples by baseline success Cui & Chen (2026). MeanDiff represents this partition by one displacement between class means, whereas ULRM learns an unpaired nonlinear map between the two regions. Classic CAA uses stronger supervision by constructing its direction from paired preferred and nonpreferred completions.

Lower-supervision representation discovery. Other interpretability work studies whether useful directions can be recovered from weaker signals, including unlabeled activations and consistency structure Burns et al. (2024). This line suggests that internal representations can contain behavior-relevant information even when explicit labels or counterfactual completions are unavailable. ULRM brings the same low-supervision motivation to activation steering: instead of learning a single direction from paired demonstrations, it treats preferred and nonpreferred decision states as empirical activation regions and learns a map between them.

3 Method

Behavior split. Let X be a dataset for a target behavior. We evaluate the model zero-shot and split examples into X^+, where the model exhibits the preferred behavior, and X^-, where it does not. In the A/B setting this partition is induced by the model's own answer choice, not by constructing a paired preferred and dispreferred completion for each prompt. For an example x and layer ℓ, let $h_\ell(x)$ denote the model's hidden state at the prompt-side decision position, i.e., the activation used immediately before scoring the answer token. We collect these decision activations and obtain finite sets $S^+ = \{h_\ell(x) : x \in X^+\}$ and $S^- = \{h_\ell(x) : x \in X^-\}$. We interpret these as samples from latent regions H^+ and H^-.

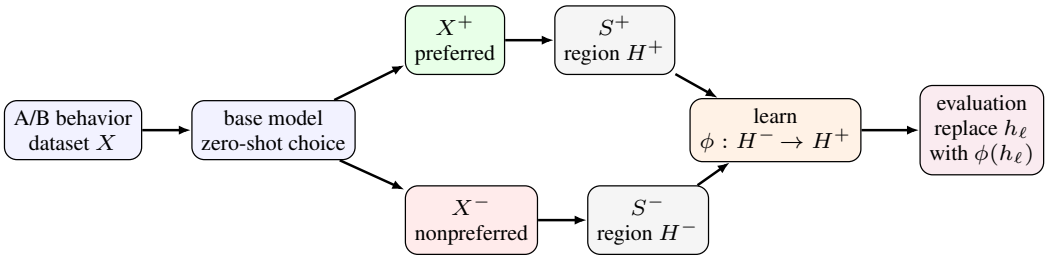

Figure 1: Overview of ULRM. We use observed model behavior to define preferred and nonpreferred activation regions, learn a map toward the preferred region, and apply it at the prompt-side decision state.

Unsupervised Latent Region Matching. We train one multilayer perceptron ϕ that moves activations from H^- toward H^+. To preserve information already compatible with the preferred behavior, the same map remains approximately identity on H^+. For balanced minibatches $B^- = B \cap S^-$ and $B^+ = B \cap S^+$, we optimize

$$\mathcal{L}(B) = \frac{\alpha}{|B^+|} \sum_{x \in B^+} \frac{\|x - \phi(x)\|_2^2}{d_h} + (1-\alpha)\mathcal{D}_{H^+}\big(\phi(B^-)\big)\,, \tag{1}$$

where d_h is the hidden-state dimensionality, $\phi(B^-) = \{\phi(x) : x \in B^-\}$, and $\mathcal{D}_{H^+}$ is an empirical discrepancy from the successful activation region. The identity term anchors the map on states already in its target region, while the region term pulls unsuccessful states toward observed successful activations.

The choice of region discrepancy gives two distinct methods, which we evaluate separately: k-nearest-neighbor ULRM (kNN-ULRM) and maximum mean discrepancy ULRM (MMD-ULRM). They share the behavior split, mapper architecture, identity term, and training procedure. Only the representation of the target region differs. kNN-ULRM applies a local pull toward nearby successful activations, whereas MMD-ULRM aligns the distribution of a mapped unsuccessful minibatch with the full empirical successful region.

kNN-ULRM measures how far each mapped activation is from its k nearest successful activations, then averages this distance over the minibatch. For a mapped activation z, let $u_1, \ldots, u_k$ be its nearest successful activations:

$$\mathcal{D}_{H^+}^{\mathrm{kNN}}(z) = \sum_{j=1}^{k} w_j \|z - u_j\|_2, \qquad w_j = \frac{\exp(-\|z-u_j\|_2/\tau)}{\sum_r \exp(-\|z-u_r\|_2/\tau)}. \tag{2}$$

The weights give more importance to closer neighbors. MMD-ULRM instead compares the mapped minibatch with the full set of successful activations. Let $Z = \{z_i\}_{i=1}^m = \phi(B^-)$ be the mapped minibatch and $U = \{u_j\}_{j=1}^n = S^+$ the successful activation set. We use squared maximum mean discrepancy (MMD) Gretton et al. (2012):

$$\begin{aligned}\mathcal{D}_{H^+}^{\mathrm{MMD}}(Z) = \frac{1}{m^2}\sum_{i,i'=1}^{m} k(z_i, z_{i'}) + \frac{1}{n^2}\sum_{j,j'=1}^{n} k(u_j, u_{j'}) \\ - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(z_i, u_j),\end{aligned} \tag{3}$$

$$k(a,b) = \frac{1}{|\mathcal{C}|}\sum_{c \in \mathcal{C}} \exp\left(-\frac{\|a-b\|_2^2}{2c^2\rho^2}\right).$$

Here $\mathcal{C} = \{0.25, 0.5, 1, 2, 4\}$ contains the kernel scales, and ρ^2 is the median positive squared distance between pairs in U. Because the first two sums include self-pairs, this is the biased MMD estimator. In short, kNN-ULRM pulls each activation toward nearby successful examples, while MMD-ULRM matches the mapped and successful distributions. In both methods, identity regularization keeps already-successful activations nearly unchanged. Figure 2 visualizes one learned map in a fixed PCA projection.

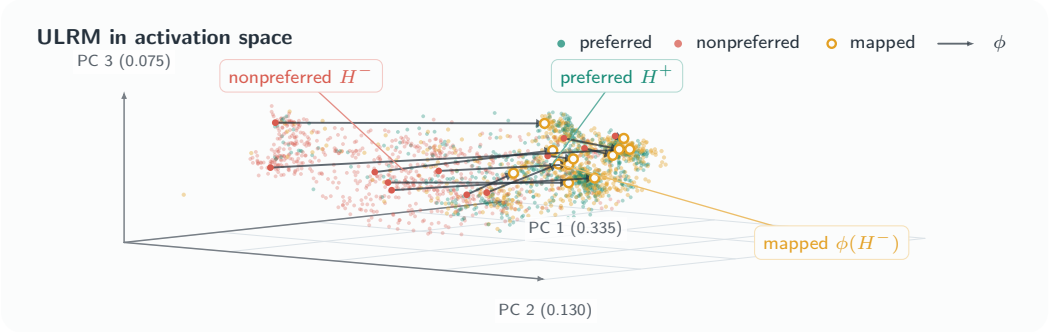

Figure 2: PCA view of the accuracy-selected MMD-ULRM mapper for Llama/PAS at layer 31. PCA is fitted only to the original training activations, and mapped states use the same fixed basis. Points show fixed-seed samples of 700 states per cloud. Arrows show 12 randomly selected mappings. The three PCs capture 0.540 of the original variance, while ULRM operates in all 4,096 dimensions.

Intervention. At inference time, we apply ϕ to the same prompt-side decision-state activation used by the A/B evaluation. The resulting activation replaces the original hidden state at layer ℓ before the model scores the answer token. This keeps training and evaluation aligned: the mapper is trained on decision-state activations and applied at the same kind of position. No language-model weights are updated. Figure 1 summarizes the full ULRM pipeline.

4 Experimental Setup

Datasets. We use the model-written sycophancy evaluations introduced by Perez et al. (2022). Each example presents a user profile and viewpoint followed by a two-choice question. We evaluate the data selections used by PAS Cui & Chen (2026) and CAA Panickssery et al. (2024), containing 30,051 and 20,184 examples, respectively.

The source prompts contain embedded choices such as "(A) Agree" and "(B) Disagree." We parse these choices and present their semantic content directly, for example `A: Disagree` and `B: Agree`, with deterministic answer-order shuffling. The preferred target is always the anti-sycophantic answer that does not match the user's stated belief. Each model is prompted through its tokenizer-native instruction template without a custom system message, and we evaluate its next-token probabilities for the two answer labels.

For each model and pool, we first evaluate the baseline on the full source. We downsample the larger of the baseline-correct and baseline-wrong groups, then form train, validation, and test splits that are each exactly 50/50. Validation contains 200 examples, test contains 500, and training uses all remaining balanced pairs up to 5,000 examples. The resulting train sizes are 4,514 and 3,852 for Llama on PAS and CAA, and 5,000 and 4,820 for Qwen. The balanced evaluation tests whether steering preserves correct answers and fixes incorrect ones, rather than measuring performance on the original data distribution.

Models and training. All experiments use Llama-3.1-8B-Instruct Grattafiori et al. (2024) and Qwen2.5-7B-Instruct Qwen et al. (2025). We collect prompt-side activations and sweep zero-based decoder blocks 13–31 for Llama and 11–27 for Qwen. Vector methods scan these full ranges. For each ULRM method, model, and pool, we select one layer by validation full-vocabulary accuracy in a preliminary layer sweep. We reuse that layer for the final run and ablations. At that layer, checkpoints are selected separately for each reporting metric. The ULRM mapper is a four-hidden-layer, width-2048 GELU residual MLP trained for 2,000 epochs, with checkpoints saved every 100 epochs. Appendix A gives the remaining optimization and objective hyperparameters.

Data-scaling protocol. We additionally evaluate nested, balanced training sets of 156, 312, 624, 1,250, and 2,500 examples, followed by the balanced training set used in each main-experiment setting. We keep the validation and test sets unchanged and fix every method's layer to its main-experiment choice, separately by metric where the original choices differ. To separate data quantity

Table 1: Test results. Each cell is $\mathrm{Acc}_{\mathrm{vocab}} \,/\, \overline{p}_{\mathrm{correct}}$, reported as a proportion. For MeanDiff and CAA, validation selects a layer–strength pair separately for each metric. For each ULRM method, model, and pool, a preliminary validation-accuracy sweep selects one layer shared by both metrics, while validation selects a checkpoint separately for each metric. Bold marks the best and underlining the second-best value for each metric in each row. "Average" is the arithmetic mean over the four model–pool settings.

Model	Pool	Base	MeanDiff	CAA	kNN-ULRM	MMD-ULRM
Llama	PAS	0.500/0.391	0.538/0.467	0.492/0.423	<u>0.608</u>/<u>0.490</u>	**0.674/0.529**
Llama	CAA	0.500/0.390	<u>0.580</u>/<u>0.522</u>	0.496/0.407	0.528/0.414	**0.632/0.525**
Qwen	PAS	0.494/0.476	0.488/0.478	0.524/0.517	<u>0.526</u>/<u>0.526</u>	**0.552/0.532**
Qwen	CAA	0.498/0.481	0.500/0.481	0.498/0.481	<u>0.518</u>/<u>0.515</u>	**0.538/0.524**
Average		0.498/0.435	0.527/<u>0.487</u>	0.503/0.457	<u>0.545</u>/0.486	**0.599/0.528**

from optimization time, ULRM uses 36k, 32k, 40k, and 38k optimizer updates for Llama/PAS, Llama/CAA, Qwen/PAS, and Qwen/CAA, respectively, at every training-set size. Validation selects among 20 evenly spaced checkpoints. We also train on all remaining examples without balancing the dataset: 29,351 examples for PAS and 19,484 for CAA. ULRM still draws balanced minibatches for its two loss terms.

Baselines and metrics. We compare both ULRM methods with the unsteered model and two vector baselines. MeanDiff uses the same unpaired success/failure training activations as ULRM. Its vector is $v_{\mathrm{MD}} = \mathbb{E}_{x \in S^+_{\mathrm{tr}}}[x] - \mathbb{E}_{x \in S^-_{\mathrm{tr}}}[x]$, the difference between the successful and unsuccessful training means. Classic CAA Panickssery et al. (2024) instead uses a gold pair for every training prompt x_i: preferred answer text a_i^+ and nonpreferred answer text a_i^-. Its vector is $v_{\mathrm{CAA}} = N^{-1} \sum_i [h_\ell(x_i, a_i^+) - h_\ell(x_i, a_i^-)]$, where $h_\ell(x_i, a)$ is the activation at the final token of answer completion a. At evaluation, both methods add sv to the prompt-side decision activation, where s is a scalar steering strength that controls the magnitude of the intervention. We do not normalize the vectors, and sweep $s \in \{.5, 1, 2, 4, 8, 16, 32\}$.

The A/B datasets identify a preferred answer token for each example. Let $a_i^\star$ be that token for example i, and let $p_i(a)$ be its next-token probability over the full vocabulary V. We report

$$\mathrm{Acc}_{\mathrm{vocab}} = \frac{1}{n} \sum_i \mathbf{1}\left[a_i^\star = \arg\max_{a \in V} p_i(a)\right], \qquad \overline{p}_{\mathrm{correct}} = \frac{1}{n} \sum_i p_i(a_i^\star). \tag{4}$$

Full-vocabulary accuracy is deliberately stricter than selecting the more likely of A and B: the correct letter must be the model's global next-token argmax. Mean correct-token probability retains graded information even when another vocabulary token is most likely. Both metrics lie in $[0, 1]$ and are reported as proportions.

For vector methods, every layer–strength pair produces both validation metrics. Layer and strength are selected independently per metric. For ULRM, all 20 checkpoints at the fixed layer produce both validation metrics, and the checkpoint is selected independently per metric. We report only each metric's matching test result.

5 Results

Table 1 shows that behavior-split latent-region matching produces substantial steering signal from a remarkably limited source of supervision. MMD-ULRM uses only the model's observed preferred and nonpreferred decisions to form activation regions, yet raises average full-vocabulary accuracy from 0.498 to 0.599 and mean correct-token probability from 0.435 to 0.528. Compared with MeanDiff, which uses the same behavior split, these are absolute gains of 0.072 and 0.041, suggesting that the nonlinear map and geometric matching objective add useful steering capacity beyond the partition itself.

The comparison to CAA is intentionally demanding. CAA receives paired contrastive completions that directly identify a steering direction, whereas ULRM receives only an unpaired behavior

Table 2: Fixed-layer objective and architecture ablations on the test sets. Each cell is $\mathrm{Acc}_{\mathrm{vocab}} / \overline{p}_{\mathrm{correct}}$, reported as a proportion. Full is the standard ULRM configuration from Equation 1, $-\mathcal{L}_{\mathrm{id}}$ removes identity preservation, and Linear replaces the standard nonlinear mapper with a residual affine mapper. Checkpoints are selected independently on validation for each metric. Bold marks the best and underlining the second-best value for each metric in each row.

Model	Pool	Method	Full	$-\mathcal{L}_{\mathrm{id}}$	Linear
Llama	PAS	KNN	**0.608/0.490**	0.566/0.463	0.478/0.418
Llama	PAS	MMD	**0.674/0.529**	0.586/0.484	0.560/0.507
Llama	CAA	KNN	**0.528/0.414**	0.506/**0.414**	0.430/0.397
Llama	CAA	MMD	**0.632/0.525**	0.628/0.504	0.546/0.480
Qwen	PAS	KNN	**0.526/0.526**	0.504/0.497	0.478/0.474
Qwen	PAS	MMD	**0.552/0.532**	0.524/0.512	0.502/0.498
Qwen	CAA	KNN	0.518/0.515	**0.544/0.534**	0.516/0.513
Qwen	CAA	MMD	0.538/0.524	**0.576/0.568**	0.500/0.489
KNN average			**0.545/0.486**	0.530/0.477	0.476/0.451
MMD average			**0.599/0.528**	0.579/0.517	0.527/0.494

split. Under this supervision gap, MMD-ULRM is numerically above both vector baselines in every model–pool–metric cell. Its strongest result is 0.674 Llama/PAS accuracy, versus 0.538 for MeanDiff, 0.492 for CAA, and 0.500 unsteered.

KNN-ULRM is less consistent: it improves on the unsteered model in all four settings, but trails MeanDiff on Llama/CAA and is essentially tied with MeanDiff in average probability. These results show that kNN and MMD matching behave differently.

6 Ablations and Data Sensitivity

We next ask which design choices support ULRM's performance and how much data those choices require. We first ablate identity preservation and mapper nonlinearity, then vary the number of balanced training examples, and finally compare balanced subsets with all available training data. These experiments distinguish sensitivity to the intervention's design from sensitivity to the data used to estimate the activation regions.

6.1 Objective and mapper architecture

We ablate the objective and mapper architecture at the layer used for each main result. The first variant removes identity preservation, leaving only region matching. The second retains the full objective but replaces the four-hidden-layer MLP with a residual affine map, $\phi(x) = x + Wx + b$. Checkpoints are selected independently on validation for each metric.

Table 2 shows that removing identity preservation lowers average accuracy by approximately 1.5 percentage points for KNN and 2.1 points for MMD, and lowers mean correct-token probability by 0.009 and 0.011, respectively. Identity preservation helps on average, but removing it improves both metrics for both methods on Qwen/CAA. Replacing the nonlinear mapper with a residual affine map lowers average accuracy by approximately 7.0 percentage points for KNN and 7.2 points for MMD, and lowers mean correct-token probability by 0.036 and 0.034. Full ULRM outperforms the affine variant in every model–pool–method comparison on both metrics. These differences are computed from the displayed per-setting values before rounding the averages.

6.2 How much training data is needed?

Learning a nonlinear map between two empirical activation regions may require more examples than estimating a single steering vector. We therefore ask whether ULRM's advantage persists when fewer successful and unsuccessful states are available. This tests the data requirement of region matching, which matters for behaviors where collecting a large training set is difficult.

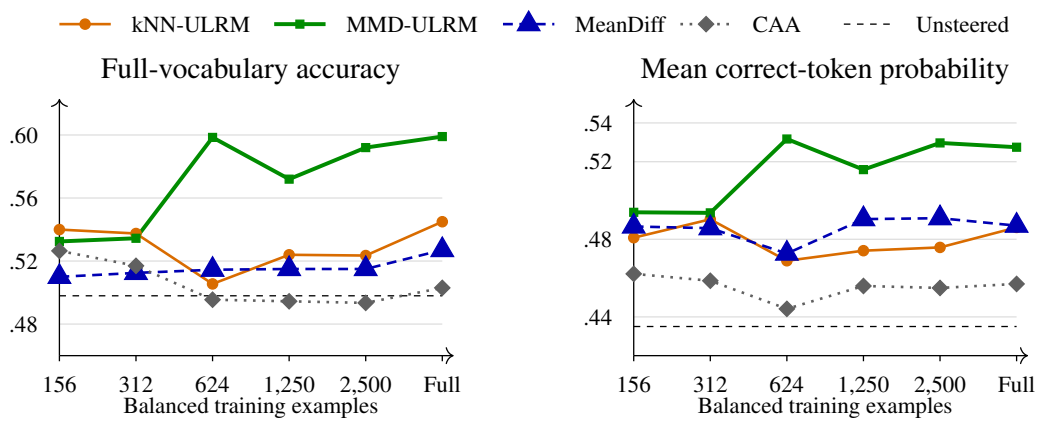

Figure 3: Test performance as balanced training-set size increases, averaged over the four model–pool settings. "Full" uses each setting's largest balanced set (3,852–5,000 examples). Layers and ULRM update budgets are fixed, while validation selects a checkpoint or strength separately for each metric. ULRM full-set points are exact-step reruns. MeanDiff and CAA reuse the endpoints in Table 1. Curves show one seed and are not uncertainty estimates.

We train on nested, balanced subsets ranging from 156 examples to the largest capped balanced set in each setting. Within each model–pool setting, we hold the intervention layer, validation and test sets, and ULRM optimizer-step budget fixed, and select checkpoints or vector strengths on validation. Matching optimizer updates prevents larger datasets from receiving more training steps simply because an epoch contains more batches. The layers were selected in the main experiments, so this study measures sensitivity to the number of training examples conditional on those layer choices. It does not measure the total data needed to select an intervention from scratch.

Figure 3 shows that MMD-ULRM improves sharply between 312 and 624 examples: its average accuracy rises from 0.535 to 0.599 and its average correct-token probability from 0.494 to 0.532. It remains the strongest method on average on both metrics at every larger size. At the largest balanced sizes, it reaches 0.599/0.528, compared with 0.545/0.486 for kNN-ULRM, 0.527/0.487 for MeanDiff, and 0.503/0.457 for CAA. The curves are not monotonic, and each uses one seed: 624 is the smallest tested size with an average lead on both metrics, rather than an established minimum data requirement.

6.3 Does retaining all available training data help?

The balanced experiments discard examples from the larger baseline-outcome group. This leaves a practical question: does retaining these examples improve the estimate of the activation regions, or does changing the composition of the training set weaken the intervention? The comparison matters because baseline success and failure need not be equally common in an available behavior dataset.

We compare the capped balanced sets with all remaining source examples after removing the same validation and test sets: 29,351 training examples for PAS and 19,484 for CAA. Layers and ULRM update budgets remain fixed. Although the full datasets retain their natural class proportions, ULRM still samples balanced minibatches. "Natural" therefore describes the available training pool, not ULRM's minibatch distribution. Evaluation remains on the balanced test sets, so this experiment tests the effect of retaining more training data rather than performance at the source distribution's natural prevalence.

Using every available training example does not uniformly improve steering (Table 3). MMD-ULRM remains strongest on average at 0.588/0.520, close to its matched-update capped-balanced result of 0.599/0.528. CAA falls from 0.503/0.457 on the balanced sets to 0.480/0.432 on the natural training distributions, below the unsteered average of 0.498/0.435.

The natural and balanced sets differ in both size and class proportions, so this comparison does not isolate the effect of class imbalance.

Table 3: Full-data test results. Each cell is $\mathrm{Acc}_{\mathrm{vocab}} \,/\, \overline{p}_{\mathrm{correct}}$. "Natural" retains all source training examples after removing the fixed validation and test sets. ULRM still draws balanced minibatches. Bold marks the best and underlining the second-best value for each metric in each row.

Model	Pool	Train set	n	Base	kNN-ULRM	MMD-ULRM	MeanDiff	CAA
Llama	PAS	Natural	29,351	0.500/0.391	**0.606**/0.488	0.590/**0.489**	0.538/0.467	0.484/0.399
Llama	CAA	Natural	19,484	0.500/0.390	0.526/0.418	**0.664/0.532**	0.580/0.525	0.464/0.380
Qwen	PAS	Natural	29,351	0.494/0.476	0.520/0.513	**0.556/0.538**	0.488/0.477	0.486/0.475
Qwen	CAA	Natural	19,484	0.498/0.481	0.530/0.520	**0.542/0.522**	0.498/0.481	0.484/0.476
Natural average				0.498/0.435	0.546/0.485	**0.588/0.520**	0.526/0.488	0.480/0.432

Table 4: Computational cost of the complete two-model, two-pool search in NVIDIA H100 GPU-hours and FLOPs. Activation collection means extracting the hidden states used by each method.

Method	Activation collection	Mapper training	Validation search	Total	FLOPs
KNN-ULRM	0.71	30.77	13.25	44.88	3.899 EF
MMD-ULRM	0.71	30.67	13.24	44.78	3.899 EF
MeanDiff	0.71	0.00	4.64	5.50	2.033 EF
Classic CAA	7.39	0.00	4.61	12.16	3.074 EF

7 Computational tradeoff

ULRM's greater flexibility comes from training a separate mapper, which adds a cost beyond estimating a steering vector. We therefore compare the computation needed to collect activations, construct the intervention, and select its configuration on validation. Table 4 reports GPU time and FLOPs for the complete search across both models and both data pools.

Table 4 shows that ULRM requires more computation because it trains a nonlinear mapper and validates learned checkpoints. KNN-ULRM and MMD-ULRM each require about 45 NVIDIA H100 GPU-hours and 3.899 EFLOP. MMD-ULRM takes 8.1$\times$ the GPU-hours of MeanDiff and 3.7$\times$ those of CAA.

The cost breakdown shows where this difference arises. ULRM and MeanDiff use the same type of prompt-side decision activations, with a collection cost of 0.71 GPU-hours each. CAA instead collects paired completion activations, costing 7.39 GPU-hours under our protocol. However, ULRM's lower collection cost is outweighed by roughly 31 GPU-hours of mapper training and 13 GPU-hours of validation search. The two vector baselines require no mapper training and spend about 4.6 GPU-hours on validation.

MeanDiff is therefore the least expensive option in this comparison, while MMD-ULRM trades additional training and selection cost for higher scores on the evaluated tasks. The two ULRM variants have similar computational costs, so MMD's stronger average results do not come with a substantial increase in the reported search budget over kNN. These costs describe the search procedure. At inference, ULRM evaluates the selected mapper, whereas the vector baselines add a fixed displacement. We do not measure the resulting latency difference here.

8 Conclusion

We introduced ULRM, a behavior-split latent-region matching approach with two methods, KNN-ULRM and MMD-ULRM, for activation steering without paired contrastive completions. On two sycophancy pools and two instruction models, MMD-ULRM improves both metrics over the unsteered model, MeanDiff, and CAA on average while using less supervision than CAA. Its advantage appears by 624 balanced training examples and persists at larger sizes. MMD-ULRM also remains strongest on the full natural training distributions, while CAA is more sensitive to data composition. Together, these results show that naturally occurring preferred and nonpreferred decisions contain

useful steering information and that matching their activation distributions can outperform a single global direction.

9 LIMITATIONS

There are several limitations and opportunities for improvement. We study two sycophancy pools, two 7–8B models, one split, and one seed. Balanced test sets evaluate whether an intervention preserves baseline-correct answers and fixes baseline-wrong answers, rather than accuracy at the source data's natural prevalence. Independent multi-seed replication with broader behaviors and model families is therefore needed.

ULRM is less immediately interpretable than a single steering vector. A learned nonlinear map can represent local variation across examples, and understanding this local structure is an important mechanistic opportunity. Finally, this paper studies single-token A/B decisions only. Open-ended generation, multi-token decisions, and stronger robustness checks remain future work.

IMPACT STATEMENT

ULRM may improve model interactions by reducing unwanted sycophancy without modifying model weights, but the same mechanism could reinforce deceptive or manipulative behavior. Evaluation on one behavior and an MCQ format does not establish deployment safety, so practical use requires broader testing, monitoring, and an unsteered fallback.

AI USE STATEMENT

We used OpenAI Codex to polish and organize the manuscript and obtain feedback on the presentation and interpretation of existing experimental results.

REFERENCES

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. URL https://arxiv.org/abs/2204.05862.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. Discovering latent knowledge in language models without supervision, 2024. URL https://arxiv.org/abs/2212.03827.

Sasha Cui and Zhongren Chen. Painless activation steering: An automated, lightweight approach for post-training large language models, 2026. URL https://arxiv.org/abs/2509.22739.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra,

Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan

Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model, 2024. URL https://arxiv.org/abs/2306.03341.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition, 2024. URL https://arxiv.org/abs/2312.06681.

Ethan Perez, Sam Ringer, Kamilė Lukošiūtė, Karina Nguyen, Edwin Chen, Scott Heiner, Craig Pettit, Catherine Olsson, Sandipan Kundu, Saurav Kadavath, Andy Jones, Anna Chen, Ben Mann, Brian Israel, Bryan Seethor, Cameron McKinnon, Christopher Olah, Da Yan, Daniela Amodei, Dario Amodei, Dawn Drain, Dustin Li, Eli Tran-Johnson, Guro Khundadze, Jackson Kernion, James Landis, Jamie Kerr, Jared Mueller, Jeeyoon Hyun, Joshua Landau, Kamal Ndousse, Landon Goldberg, Liane Lovitt, Martin Lucas, Michael Sellitto, Miranda Zhang, Neerav Kingsland, Nelson Elhage, Nicholas Joseph, Noemí Mercado, Nova DasSarma, Oliver Rausch, Robin Larson, Sam McCandlish, Scott Johnston, Shauna Kravec, Sheer El Showk, Tamera Lanham, Timothy Telleen-Lawton, Tom Brown, Tom Henighan, Tristan Hume, Yuntao Bai, Zac Hatfield-Dodds, Jack Clark, Samuel R. Bowman, Amanda Askell, Roger Grosse, Danny Hernandez, Deep Ganguli, Evan Hubinger, Nicholas Schiefer, and Jared Kaplan. Discovering language model behaviors with model-written evaluations, 2022. URL https://arxiv.org/abs/2212.09251.

Joris Postmus and Steven Abreu. Steering large language models using conceptors: Improving addition-based activation engineering, 2025. URL https://arxiv.org/abs/2410.16314.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li,

Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Nishant Subramani, Nivedita Suresh, and Matthew E. Peters. Extracting latent steering vectors from pretrained language models, 2022. URL https://arxiv.org/abs/2205.05124.

Jiuding Sun, Sidharth Baskaran, Zhengxuan Wu, Michael Sklar, Christopher Potts, and Atticus Geiger. Hypersteer: Activation steering at scale with hypernetworks, 2025. URL https://arxiv.org/abs/2506.03292.

Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering, 2024. URL https://arxiv.org/abs/2308.10248.

Zhengxuan Wu, Qinan Yu, Aryaman Arora, Christopher D. Manning, and Christopher Potts. Improved representation steering for language models, 2025. URL https://arxiv.org/abs/2505.20809.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2025. URL https://arxiv.org/abs/2310.01405.

A Reproducibility details

Runtime. Each experiment job requested one NVIDIA H100 GPU, 14 CPU cores, and 140 GiB RAM. Frozen-model scoring used batch size 8 in BF16. Decoder layer indices are zero-based. Dataset and training randomness use seed 39.

Mapper architecture. For input dimension d, the mapper is

$$\phi(h) = h + W_{\text{out}}\,\text{GELU}(W_4\,\text{GELU}(\cdots\text{GELU}(W_1 h)))),$$

with four width-2048 hidden layers, biases in every linear layer, and no dropout. The mapper has 29.37M parameters for Llama ($d = 4096$) and 27.27M for Qwen ($d = 3584$).

Training and objective hyperparameters. Mapper training uses balanced batches of 256, AdamW with learning rate 10^{-4} and weight decay 10^{-5}, BF16 autocast, and gradient clipping at 1. KNN uses $\alpha = .2$, $k = 32$, and $\tau = .1$. MMD uses $\alpha = .1$ and RBF-kernel scales $\{.25, .5, 1, 2, 4\}$.

Licenses. The sycophancy data are licensed under Creative Commons Attribution 4.0 (CC BY 4.0). Qwen2.5 is released under Apache 2.0, while Llama use is governed by the Llama 3.1 Community License.

B Training-size results

Tables 5–8 give the per-setting results underlying Figure 3. Each cell reports $\text{Acc}_{\text{vocab}} \,/\, \overline{p}_{\text{correct}}$. The reduced sets are nested: every smaller set is contained in every larger set for the same model and pool. "Full" denotes the largest capped balanced set, whose size is given in each caption. The full ULRM entries are the matched-update reruns used in Figure 3. Full MeanDiff and CAA entries are the results from Table 1.

Table 5: Training-size results for Llama/PAS. The full balanced set contains 4,514 examples.

n	kNN-ULRM	MMD-ULRM	MeanDiff	CAA
156	0.616/0.477	0.564/0.498	0.476/0.436	0.530/0.430
312	0.602/0.524	0.546/0.475	0.490/0.453	0.494/0.413
624	0.476/0.406	0.608/0.512	0.488/0.454	0.492/0.418
1,250	0.518/0.449	0.610/0.501	0.492/0.461	0.494/0.420
2,500	0.544/0.455	0.636/0.523	0.492/0.471	0.490/0.421
Full	0.608/0.490	0.674/0.529	0.538/0.467	0.492/0.423

Table 6: Training-size results for Llama/CAA. The full balanced set contains 3,852 examples.

n	kNN-ULRM	MMD-ULRM	MeanDiff	CAA
156	0.480/0.412	0.574/0.501	0.580/0.551	0.496/0.407
312	0.500/0.400	0.550/0.474	0.580/0.531	0.496/0.404
624	0.478/0.424	0.686/0.546	0.580/0.478	0.500/0.404
1,250	0.516/0.413	0.588/0.509	0.580/0.541	0.496/0.406
2,500	0.500/0.412	0.630/0.516	0.580/0.532	0.498/0.404
Full	0.528/0.414	0.632/0.525	0.580/0.522	0.496/0.407

Table 7: Training-size results for Qwen/PAS. The full capped balanced set contains 5,000 examples.

n	kNN-ULRM	MMD-ULRM	MeanDiff	CAA
156	0.554/0.536	0.480/0.474	0.488/0.478	0.590/0.532
312	0.542/0.531	0.498/0.488	0.484/0.478	0.578/0.536
624	0.548/0.530	0.530/0.510	0.490/0.478	0.490/0.477
1,250	0.540/0.518	0.534/0.510	0.488/0.479	0.490/0.518
2,500	0.540/0.531	0.534/0.533	0.488/0.479	0.490/0.515
Full	0.526/0.526	0.552/0.532	0.488/0.478	0.524/0.517

Table 8: Training-size results for Qwen/CAA. The full balanced set contains 4,820 examples.

n	kNN-ULRM	MMD-ULRM	MeanDiff	CAA
156	0.510/0.498	0.512/0.503	0.496/0.481	0.490/0.480
312	0.506/0.506	0.544/0.537	0.496/0.481	0.500/0.481
624	0.520/0.515	0.570/0.558	0.500/0.481	0.500/0.478
1,250	0.522/0.516	0.556/0.544	0.500/0.481	0.498/0.478
2,500	0.510/0.505	0.568/0.548	0.500/0.481	0.496/0.480
Full	0.518/0.515	0.538/0.524	0.500/0.481	0.498/0.481

C DATA-SCALING SELECTIONS

Table 9 reports the validation-selected candidate for every balanced data-scaling run. Each entry gives the candidate selected for accuracy followed by the candidate selected for probability. ULRM entries are optimizer steps, while MeanDiff and CAA entries are steering strengths. The fixed kNN/MMD layers are 30/31, 16/29, 27/22, and 22/27 for Llama/PAS, Llama/CAA, Qwen/PAS, and Qwen/CAA, respectively. MeanDiff uses accuracy/probability layers 30/18, 30/30, 26/12, and 12/13. CAA uses layers 13, 13, 15, and 20.

Table 9: Validation-selected ULRM steps and vector strengths in the balanced data-scaling study. Entries are accuracy/probability selections.

Setting	n	kNN step	MMD step	MeanDiff s	CAA s
Llama/PAS	156	7200/30600	32400/1800	.5/4	4/2
	312	1800/14400	10800/28800	.5/8	1/2
	624	34200/36000	9000/34200	.5/8	1/2
	1,250	16200/36000	28800/34200	.5/8	1/2
	2,500	12600/32400	30600/16200	.5/8	1/2
	Full	23400/36000	32400/27000	32/8	4/2
Llama/CAA	156	24000/24000	25600/11200	16/16	.5/2
	312	25600/30400	4800/9600	16/16	.5/1
	624	14400/30400	27200/27200	32/16	.5/1
	1,250	1600/1600	12800/16000	16/16	.5/2
	2,500	30400/1600	16000/16000	16/16	.5/1
	Full	30400/1600	8000/9600	16/16	.5/2
Qwen/PAS	156	22000/22000	18000/36000	.5/4	16/16
	312	34000/2000	22000/22000	.5/4	16/16
	624	10000/4000	14000/14000	.5/4	.5/.5
	1,250	18000/4000	14000/10000	.5/4	.5/16
	2,500	10000/10000	40000/34000	.5/4	.5/16
	Full	14000/14000	38000/38000	.5/2	16/16
Qwen/CAA	156	7600/7600	1900/3800	.5/1	2/1
	312	5700/3800	17100/38000	.5/2	.5/.5
	624	15200/1900	13300/13300	1/.5	.5/2
	1,250	22800/20900	38000/38000	.5/.5	.5/2
	2,500	3800/3800	36100/36100	1/.5	.5/1
	Full	7600/11400	9500/9500	1/1	.5/1

Table 10: Validation-selected candidates for the full-data experiments. Entries are accuracy/probability selections. ULRM entries are optimizer steps. Vector entries are strengths. Layers are the same fixed layers listed above.

Model	Pool	Train set	kNN step	MMD step	MeanDiff s	CAA s
Llama	PAS	Natural	16200/36000	7200/7200	16/8	.5/.5
Llama	CAA	Natural	30400/3200	19200/4800	16/16	.5/.5
Qwen	PAS	Natural	2000/2000	38000/26000	.5/2	.5/.5
Qwen	CAA	Natural	1900/1900	17100/17100	.5/.5	.5/.5

D SELECTED CONFIGURATIONS

Table 11: Reported configurations. ULRM entries are fixed `layer/validation-selected epoch`. Vector entries are validation-selected `layer/strength`. Candidate selection is independent per metric, but the ULRM layer is shared across metrics.

Model/pool	Metric	KNN-ULRM	MMD-ULRM	MeanDiff	CAA
Llama/PAS	Accuracy	30/1200	31/1700	30/32	13/4
Llama/PAS	Probability	30/2000	31/100	18/8	13/2
Llama/CAA	Accuracy	16/1300	29/600	30/16	13/.5
Llama/CAA	Probability	16/100	29/600	30/16	13/2
Qwen/PAS	Accuracy	27/1300	22/1300	26/.5	15/16
Qwen/PAS	Probability	27/800	22/1300	12/2	15/16
Qwen/CAA	Accuracy	22/700	27/1600	12/1	20/.5
Qwen/CAA	Probability	22/700	27/2000	13/1	20/1