Analyzing and Steering Reflection in Large Reasoning Models: A Difficulty-Aware Geometric Approach

Ge Teng[1]*, Chen Shen[2], Ting Mao[2], Xiang Tian[1,3], Xuesong Liu[1,3], Rongxin Jiang[1,3], Yaowu Chen[1,3], Jieping Ye[2]

[1]Zhejiang University, [2]Token Foundry, Alibaba Group
[3]Zhejiang University Embedded System Engineering Research Center, Ministry of Education of China

Correspondence: jason.sc@alibaba-inc.com (Chen Shen), tianx@zju.edu.cn (Xiang Tian)

Abstract

Large Reasoning Models (LRMs) have demonstrated exceptional capabilities in complex problem-solving, largely driven by emergent reflection and backtracking mechanisms. However, understanding these mechanisms through mechanistic interpretability remains challenging. Current sparsity-based approaches rely heavily on heuristic keywords (e.g., "wait") and focus on isolated features, lacking a holistic view of how these features collaborate. In this paper, we propose a novel interpretability framework that bridges the microscopic and macroscopic analysis of Sparse Auto-Encoder (SAE) features. By leveraging an auxiliary LRM to robustly annotate reasoning trajectories, we introduce a comprehensive "Reflection Score" to quantify a feature's relevance to reflection without keyword bias. Our geometric analysis of these features reveals that LRMs encode reflection not as discrete, isolated vectors, but within continuous regions of the hidden state space. Furthermore, we uncover an intrinsic, difficulty-aware reflection representation mechanism: the model dynamically activates sub-regions based on problem complexity. Building upon these insights, we introduce a static, difficulty-informed steering strategy that suppresses superficial, simple-problem oriented reflection features during inference. Comprehensive evaluations across three LRMs and challenging reasoning benchmarks demonstrate that our intervention simultaneously improves reasoning accuracy by an average of 2.25% and reduces token consumption by 5.3%. Our findings provide a deeper geometric understanding of LRM reflection and offer an interpretable, scalable approach to optimizing LRM reasoning.

1 Introduction

Since their emergence in 2024, Large Reasoning Models (LRMs) (Xu et al., 2025; Zhang et al., 2025a) have demonstrated remarkable capabilities in solving complex, multi-step reasoning tasks across mathematics, science, and coding. A key driver behind the success of these Large Language Model (LLM) variants is the emergence of "reflection" and "backtracking" mechanisms, i.e., the model learned the ability to self-verify, critique, and correct intermediate outputs during generation. Such capabilities were largely absent in prior LLMs (Huang et al., 2024; Kamoi et al., 2024) and typically emerged during the Reinforcement Learning (RL) phase of LRM training (Guo et al., 2025).

As a foundational capability, the reflection mechanism (hereafter encompassing both reflection and backtracking) has naturally become a focal point for researchers. Existing efforts to analyze and optimize this mechanism include entropy analysis (Mu et al., 2025; Cai et al., 2026; Li et al., 2026), and training dynamics analysis (Yeo et al., 2025; Cheng et al., 2025). While these approaches offer a macroscopic perspective on the origins and mechanisms of reflection, they lack a microscopic view into the internal structures of the models. Consequently, mechanistic interpretability approaches have been introduced to probe these internal states. Notably, sparsity-based dictionary learning methods, such as Sparse Auto-Encoders (SAEs) (Cunningham et al., 2023) and Transcoders (Dunefsky et al., 2024), have emerged as highly promising techniques (Galichin et al., 2025; Zhang et al., 2025b). These methods project the highly entangled, dense hidden states of language models into a higher-dimensional, sparse latent space, enabling the extraction of monosemantic features—directions in the activation space corresponding to distinct, human-interpretable concepts. These localized features provide a valuable microscopic lens for dissecting the complex reflection dynamics of LRMs.

Despite these methodological advancements, existing interpretability analyses of LRM reflection face two significant limitations. First, investiga-

*Work done during a research internship at Token Foundry, Alibaba Group.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 6891–6915
October 24-29, 2026 ©2026 Association for Computational Linguistics

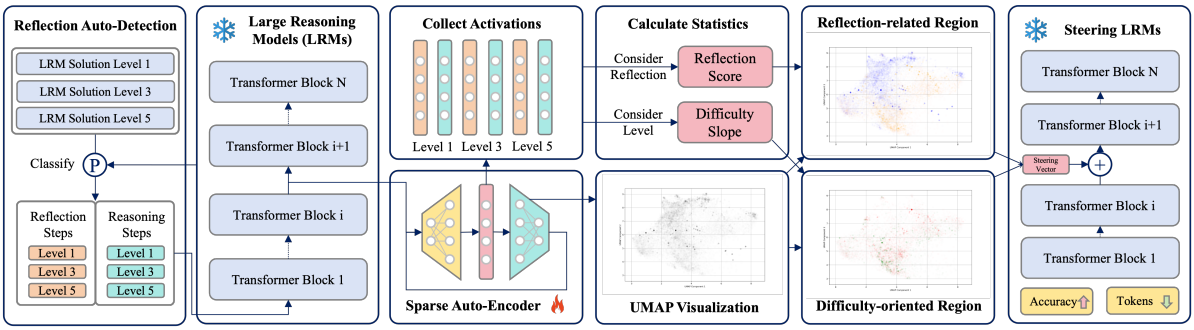


Figure 1: The overall pipeline of our approach. We first classify LRM solutions into reflection or reasoning steps with an auxiliary LRM. Then we collect SAE activations on these steps, and calculate the Reflection Score metric that represents the correlation between each SAE feature and the reflection concept, and the Difficulty Slope metric that represents the activation orientation of each SAE feature on problems of different difficulty. Finally, we combine the two metrics to calculate the a steering vector that suppress the reflection region that tend to activate on easy problems, and generally improve the performance of LRMs.

tions into reflection are frequently restricted to single-token analysis and rely heavily on heuristic keywords, anchoring their scope to specific trigger phrases (e.g., "wait", "hmm", or "but") (Qian et al., 2025; Wang et al., 2025; Galichin et al., 2025). This rule-based approach suffers from poor scalability to novel reflection patterns (e.g., newer models like DeepSeek-R1-0528 (Guo et al., 2025) and GPT-oss (OpenAI et al., 2025) rely less on explicit "wait" tokens) and introduces substantial bias, failing to capture the entirety of the reflection process. Second, while current sparsity-based analyses excel at isolating individual monosemantic features, subsequent analysis and optimization (steering) are often constrained to these isolated features. The macroscopic distribution of SAE features across the hidden state space is neglected. Consequently, the field lacks a holistic understanding of how LRM represent high-level concepts (e.g., reflection) geometrically in its latent space.

To address these limitations, we propose a novel interpretability framework (Figure 1) that bridges the microscopic and macroscopic analysis of SAE features. Overcoming the brittleness of keyword heuristics, we employ an auxiliary LRM to automatically and robustly localize reflection steps within reasoning trajectories. Based on these steps, we aggregate the activation values of each SAE feature to compute a comprehensive "Reflection Score," quantitatively measuring the correlation between individual features and the reflection concept. Utilizing the SAE feature vectors as a coordinate basis, we then systematically analyze the geometric distribution of Reflection Scores across the high-dimensional hidden state space. Our analysis reveals that LRMs encode reflection concepts through continuous regions rather than isolated feature vectors, analogous to how biological neurons form specialized brain regions for complex tasks. Furthermore, we discover that within this broader reflection region, the model dynamically activates sub-regions based on problem difficulty. Guided by this finding and the intuition that extensive reflection is often redundant for simpler problems, we design a difficulty-informed steering approach that suppresses superficial, simple-problem oriented reflection features to optimize LRM inference. Comprehensive evaluations across three LRMs and three challenging reasoning benchmarks demonstrate the effectiveness of this intervention, improving reasoning accuracy by an average of 2.25% while simultaneously reducing token consumption by 5.3%. Additional ablation studies offer further interpretability and insights into the mechanics of this reflection manipulation.

In summary, our analysis yields critical insights into LRM reflection and introduces a novel intervention strategy. Our main contributions are as follows:

- We design a robust framework to visualize and interpret the hidden state regions corresponding to specific model behaviors using SAEs. By analyzing the full context rather than relying on heuristic keywords, our approach provides a more comprehensive perspective and can easily scale to other models and behaviors.

- Leveraging this framework, we reveal that LRMs encode reflection not as isolated fea-

tures, but as continuous regions within the hidden state space. We further uncover an intrinsic, difficulty-aware mechanism, demonstrating that the specific sub-regions activated during reflection vary based on the complexity of the reasoning problem.

- Building upon these geometric insights, we introduce a difficulty-informed steering strategy. By selectively suppressing the hidden state sub-regions associated with simple-problem reflection during inference, we effectively enhance the LRM's overall performance and efficiency on highly complex reasoning tasks.

2 Find SAE features for Reflection

To address polysemanticity in Large Reasoning Models (LRMs) (Elhage et al., 2022), we employ Sparse Auto-Encoders (SAEs) (Cunningham et al., 2023) to project dense middle-layer activations $\mathbf{x} \in \mathbb{R}^{d_{model}}$ (Templeton et al., 2024) into a higher-dimensional sparse latent space $\mathbf{f} \in \mathbb{R}^{d_{SAE}}$, thereby decomposing activations into interpretable, monosemantic features. Comprehensive formalizations of the SAE architecture, objective functions, and specific configurations for the three evaluated LRMs are described in Appendix B and Table 6.

After decomposing activations into sparse features, the next challenge comes to identifying which features are related to reflection, as reflection is a high-level behavior spanning thousands of fine-grained SAE features, rendering direct auto-interpretation (Gao et al., 2025; Paulo et al., 2025) ineffective. To address this, we propose a two-step approach: using an LRM to automatically detect reflection texts, then calculating the average SAE activations on these texts to derive a comprehensive "reflection score" for each feature.

2.1 Reflection Auto-Detection with LRMs

Previous mechanistic interpretability studies typically rely on heuristic keywords (e.g., "wait" or "but") to locate reflection (Qian et al., 2025; Wang et al., 2025; Galichin et al., 2025). This rule-based approach is inherently brittle and fails to capture the full spectrum of diverse reflection representations. To overcome this, we leverage an auxiliary LRM as an automated reflection annotator.

Following prior works (Templeton et al., 2024), we first split reasoning trajectories sampled via the MATH-500 dataset (Lightman et al., 2023) into discrete steps using the "\n\n" delimiter. We then prompt DeepSeek-R1-Distill-Qwen-14B to classify each step (prompt details in Appendix C). Out of 252,298 valid steps, 41,475 (16.4%) were classified as reflection.

Method	Acc. (%)	Prec. (%)	Rec. (%)	F1 (%)
LLM Classification	**93.25**	**96.02**	91.04	**93.46**
Heuristic (Starts_With)	82.00	83.41	82.63	83.02
Heuristic (Contains)	81.75	77.38	**92.42**	84.23

Table 1: Manual verification results of reflection step detection methods across 400 randomly sampled steps. "*Starts_With*": classifying a step as reflection if it begins with a keyword. "*Contains*": classifying based on the presence of a keyword anywhere in the step. The LLM classification significantly outperforms keyword-based heuristics.

To validate the LLM annotator, we manually verified 400 random steps, comparing it against two keyword-based heuristics (using 6 common keywords from Table 7): *Starts_With* and *Contains*. As Table 1 shows, the LLM classifier achieves 93.25% accuracy and 93.46% F1, significantly outperforming heuristics, which suffer from either high false positive rates (*Contains*) or low recall (*Starts_With*). This confirms the necessity of LRM-based annotation to avoid the brittleness and bias introduced by keyword constraints.

To test whether our findings depend on a single annotator, we re-annotated the same reasoning steps with two LLMs from different families. The step labels agree at 87–88%, so our analysis does not hinge on the choice of annotator; see Appendix E for the setup and per-annotator statistics.

Table 7 in Appendix also visualized the statistics of the top first tokens of LRM identified reflection steps. While traditional keywords (e.g., wait", but", "hmm") initiate 70% of the identified reflection steps, relying solely on them would exclusively miss a significant portion of reflection behaviors, validating the necessity of our LRM-based pipeline.

2.2 Reflection Score

Building upon the automatically annotated dataset, we aim to quantitatively measure the correlation between individual SAE features and the model's reflection behavior. Inspired by recent methodologies in reasoning feature interpretation (Galichin et al., 2025), we introduce a metric termed the *Reflection Score*.

Let $\mathcal{D}_R$ denote the corpus of reflection-related tokens. In practice, to capture the contextual activation surrounding a reflection event, we define $\mathcal{D}_R$

to include the tokens within the annotated reflection steps as well as an established window (3 tokens) preceding and succeeding the step. Let $\mathcal{D}_{\neg R}$ denote the remainder of the corpus, consisting primarily of common reasoning tokens for math problems. For a given SAE feature i with activation intensity $f_i(x) \times ||\mathbf{W}^i_{dec}||$ on token x, we formulate the Reflection Score S_i as the difference in its normalized average activation between the reflection and non-reflection distributions:

$$A^R_i = \mathbb{E}_{x \in \mathcal{D}_R}[f_i(x)||\mathbf{W}^i_{dec}||] \quad (1)$$

$$A^{\neg R}_i = \mathbb{E}_{x \in \mathcal{D}_{\neg R}}[f_i(x)||\mathbf{W}^i_{dec}||] \quad (2)$$

$$S_i = d_{SAE} \cdot \left(\frac{A^R_i}{\sum_{j=1}^{d_{SAE}} A^R_j} - \frac{A^{\neg R}_i}{\sum_{j=1}^{d_{SAE}} A^{\neg R}_j} \right) \quad (3)$$

By computing S_i across all annotated steps, we successfully assigned every SAE feature with a comprehensive Reflection Score to represent its correlation with reflection concept. Crucially, by omitting variance normalization and basing S_i strictly on mean activation differences, our metric captures a diverse spectrum of reflection features. It evaluates both sparse, low-level lexical markers activating on specific tokens, and dense, abstract semantic concepts spanning multiple tokens. This deliberate design prevents vital, highly-variant sparse features from being unfairly penalized, while maintaining sensitivity to broad semantic features. We sampled and summarized several features with top reflection scores from the R1-Distill-Llama3's SAE in Table 8, Appendix D to show that this metric identifies a variety of reflection features. These features capture not only low-level lexical concepts related to specific keywords, but also higher-level, more abstract semantic concepts. These higher-order features can activate during complex self-correction and back-tracking processes, sometimes even in the absence of heuristic keywords. This confirms that the Reflection Score provides a more comprehensive and accurate view of the correlation between SAE features and the reflection concepts beyond heuristic keywords.

3 Distribution Analysis of Reflection in Hidden State Space

Prior works often focus on analyzing single feature vectors of SAEs, but they seldom turn their gaze into the correlation or distribution of these feature vectors. We all know that human brains aggregate neurons to form brain region to handle complex tasks, does the LRMs utilize their hidden state space in a similar organized manner? We find in this section that by considering the feature vectors as base vectors of the input latent space, SAEs can serve as useful tools for analyzing how LLMs represent certain concepts in their hidden state space.

3.1 Macroscopic Continuous Representation of Reflection

In the last section, we have calculated a Reflection Score for every SAE feature vector that comprehensively represents the correlation between the reflection concept and a feature direction in the LRM's hidden state space. Therefore, visualizing these feature vectors with Reflection Score provides an intuitive understanding of how reflection concepts are distributed across the LRM's hidden state space. We utilize Uniform Manifold Approximation and Projection (UMAP) to project the SAE feature vectors into a 2D space for visualization.

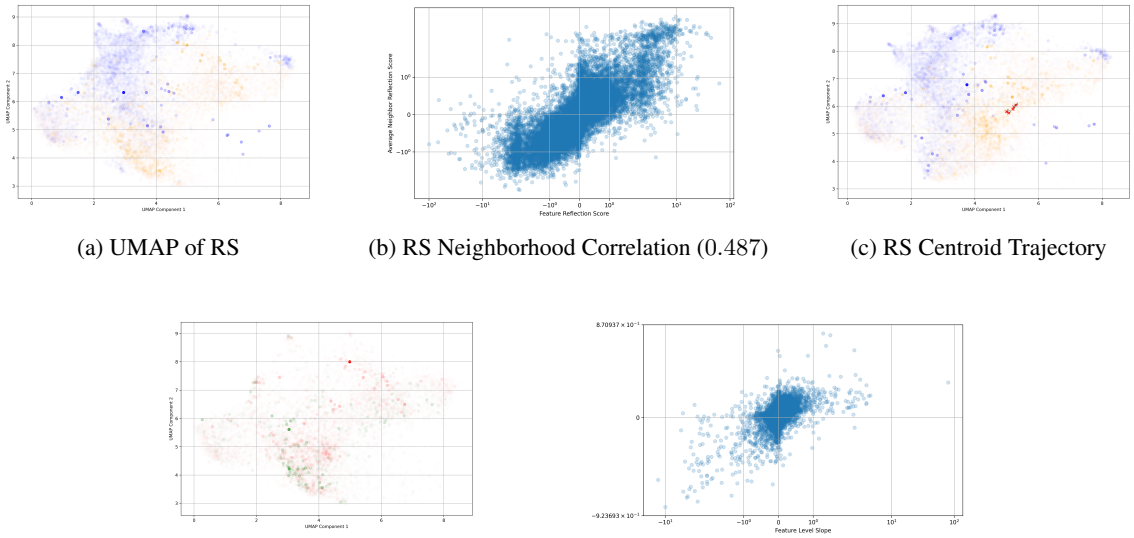
As shown in Figure 2a for the LLaMA3 model, we observe a distinct clustering phenomenon. SAE feature vectors with similar Reflection Scores aggregate in the latent space. (This observation holds across all three LRM models, as detailed in Appendix I).

To quantitatively validate this observation, we calculate the correlation between the Reflection Score of each individual feature vector and the average Reflection Score of its nearest neighbors, weighted by their cosine similarity to the feature vector. Formulated as:

$$\begin{aligned} w_{ij} &= \frac{\mathbf{W}^i_{dec} \cdot \mathbf{W}^j_{dec}}{\|\mathbf{W}^i_{dec}\| \cdot \|\mathbf{W}^j_{dec}\|}, \\ S^{Neighbor}_i &= \frac{\sum_{j \in \mathcal{K}_k} S_j \cdot w_{ij}}{\sum_{j \in \mathcal{K}_k} w_{ij}} \end{aligned} \quad (4)$$

in which $\mathcal{K}_k$ is the set of the Feature IDs of the k-nearest-neighbors of the target feature vector, and w_{ij} represents the cosine similarity serving as the weight for each neighbor. In this paper, we set $k = \sqrt{d_{SAE}}$, which is a common setting.

The results reveal a strong positive correlation, with a Pearson correlation coefficient of $r = 0.487$ for the LLaMA3 model (see Figure 2b). This demonstrates that a feature's local neighborhood effectively predicts its own Reflection Score. (Similar phenomena are observed in the other two models; see Appendix I for their visualizations). As a null baseline, randomly shuffling the Reflection



(a) UMAP of RS (b) RS Neighborhood Correlation (0.487) (c) RS Centroid Trajectory

(d) UMAP of DS (e) DS Neighborhood Correlation (0.2683)

Figure 2: Continuous representation of reflection and difficulty-aware sub-regions in DeepSeek-R1-Distill-LLaMA3-8B. Abbreviations: Reflection Score (RS), Difficulty Slope (DS). (a) UMAP visualization showing the clustering of positive (orange) and negative (blue) Reflection Scores. (b) Strong positive correlation between a feature's Reflection Score and its nearest neighbors. (c) Center of reflection feature vectors weighted by their Reflection Scores (calculated on math500 level 1-5). A clear movement from bottom-left to upper-right is observed when increasing question difficulty. (d) UMAP visualization of the Difficulty Slope. (e) Positive correlation between the Difficulty Slope of a feature and its neighbors.

Scores across features collapses this correlation to -0.001 ± 0.006 over 10 trials, confirming that it comes from the semantics attached to the feature directions rather than from the geometry of the SAE decoder alone (Appendix G). Consequently, this provides strong evidence for our early hypothesis that the model does not rely on isolated, discrete features to encode reflection. Rather, it utilizes continuous regions within the hidden state space to collectively represent the reflection concept.

3.2 Difficulty-Aware Reflection Sub-Regions

Having established the existence of a continuous latent space region dedicated to reflection, we next investigate whether this region possesses internal structure. Specifically, we ask: human brains switch and activate different brain regions to handle problems with varying complexity, does LRM utilize the hidden state reflection region in a similar manner?

To answer this, we notice that the MATH500 dataset we used are originally annotated into five difficulty levels from 1-5. We utilize this annotation and conduct preliminary visualization based on UMAP across levels, see appendix F for details. We observed a slight transition of the reflection regions as the difficulty of the problems changed.

To confirm this, we calculated and visualized the L_2-normalized center of reflection feature vectors (features with positive Reflection Score) weighted by their Reflection Score, defined as $\mathbf{C} = \tilde{\mathbf{C}}/||\tilde{\mathbf{C}}||$, where $\tilde{\mathbf{C}} = \sum_{i=1}^{d_{SAE}} \mathbb{1}_{(S_i>0)} \cdot S_i \cdot (\mathbf{W}^i_{dec}/||\mathbf{W}^i_{dec}||)$. The visualization of these centers in Figure 2c reveals a clear trajectory: a movement from the bottom left to the upper right as the difficulty level increased. Proving that when tackling problems of different difficulties, the model does not uniformly activate the entire reflection region; instead, it selectively activates or suppresses sub-regions. This indicates an intrinsic, difficulty-aware representation mechanism within the LRM's hidden state space.

To further capture the dynamics of each SAE reflection feature across problem difficulties, we introduce a novel metric: the *Difficulty Slope* of the Reflection Score. For a given feature i, let $S_i^{(l)}$ denote its Reflection Score at difficulty level $l \in \{1, 2, 3, 4, 5\}$. We fit a linear regression model $S_i^{(l)} = k_i \cdot l + b_i$ to these five data points. The slope k_i is defined as the Difficulty Slope. A positive slope ($k_i > 0$) indicates that the feature becomes

increasingly engaged in reflection as the problem difficulty rises, whereas a negative slope ($k_i < 0$) suggests the feature is primarily utilized for reflection on simpler problems.

We again apply UMAP to visualize the distribution of the Difficulty Slope k_i across the feature space, as shown in Figure 2d. Within the primary positive reflection region, a clear sub-cluster consisting of features with negative Difficulty Slope (green) can be observed for the LLaMA3 model (with similar sub-clusters present in the other LRMs, see Appendix I). This visually confirms that the model represents simple and complex reflections in different sub-regions within the broader reflection region.

To quantitatively validate this difficulty-aware clustering phenomenon, we calculate the neighborhood correlation of the Difficulty Slope, similar to our analysis of the base Reflection Scores. As shown in Figure 2e, a positive Pearson Correlation Coefficient of $r = 0.268$ is observed for the LLaMA3 model, while the corresponding shuffled control drops to 0.002 ± 0.007. This statistical evidence further confirms that features with similar difficulty orientations tend to continuously aggregate together in the latent space, and that the weaker correlation here is still not a decoder artifact.

We further verify the robustness of the Difficulty Slope: refitting it on bootstrap resamples of the problems reproduces the original values (cosine similarity 0.941 over all features), and refitting with 3, 5, or 10 difficulty bins barely changes them. Appendix H reports the confidence intervals and the binning sensitivity.

The definition above relies on the difficulty levels annotated in MATH500, which most datasets do not provide. We therefore test whether these levels can be replaced by an intrinsic signal, and find that the model's own output length recovers nearly the same slopes (Pearson $r = 0.886$). The Difficulty Slope can thus be computed, at least approximately, on datasets that carry no difficulty annotations (Appendix H).

4 Difficulty-Informed Reflection Steering

4.1 Motivation and Steering Formulation

A general consensus in the study of complex reasoning is that harder problems necessitate extensive reflection, whereas simpler problems do not. Based on this intuition, reflection behaviors that exclusively activate during simple problems are likely redundant or even counterproductive when tackling complex reasoning tasks. Therefore, suppressing these simple-problem-specific reflection features could potentially optimize the model's reasoning trajectory.

As established in Section 3, the Reflection Score (S_i) effectively distinguishes between reflection and non-reflection features, while the Difficulty Slope (k_i) successfully differentiates features that activate more on hard problems ($k_i > 0$) from those that activate more on simple problems ($k_i < 0$). To isolate the features that are both heavily involved in reflection and biased towards simple problems, we compute the element-wise product of the Reflection Score and the Difficulty Slope for each SAE feature i, and select only the subset with positive Reflection Score and negative Difficulty Slope, setting the rest to zeros:

$$w_i = \begin{cases} S_i \cdot k_i, & \text{if } S_i > 0 \text{ and } k_i < 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

The computed steering coefficient w_i finds the reflection features that tend to activate on simple problems. We first calculate the normalized steering coefficients as

$$\hat{w}_i = \alpha \cdot w_i / \sum_j |w_j| \tag{6}$$

where α is a hyper-parameter controlling the intensity of steering. Then, considering the efficiency of steering, we construct a single Steering Vector ($\mathbf{v}_{steer}$) by calculating the weighted sum of the normalized feature directions: $\mathbf{v}_{steer} = \sum_i \hat{w}_i(\mathbf{W}^i_{dec}/||\mathbf{W}^i_{dec}||)$. We apply this Steering Vector during the inference phase of LRMs by adding it directly to the corresponding residual hidden state of all token positions. By doing so, we suppress the representation of reflection features that tend to activate on simple problems. The resulting Steering Vector is robust to this selection: it is dominated by its highest-weight entries, so the features near the sign boundary, whose orientations are the least certain, barely affect it (Appendix G reports the feature counts and this sensitivity).

Difficulty information enters our method only when the steering vector is built: the Difficulty Slope is computed offline on MATH500 and decides which features to suppress. At inference time, the same fixed vector is added to every query, and the method never estimates how hard the current

Model		DeepSeek-R1-Distill-LLaMA3-8B			DeepSeek-R1-Distill-Qwen2.5-7B			Qwen3-4B-Thinking-2507		
Dataset	**Method**	**Acc. (%)↑**	**Tokens↓**	**Cutoff (%)↓**	**Acc. (%)↑**	**Tokens↓**	**Cutoff (%)↓**	**Acc. (%)↑**	**Tokens↓**	**Cutoff (%)↓**
MATH500	Baseline	87.35	5730	3.00	91.90	4215	1.05	95.55	7111	0.85
	Keyword-based	84.60 (-2.75)	**4178** (-27.1%)	**1.35** (-1.65)	88.90 (-3.00)	**3642** (-13.6%)	**0.35** (-0.70)	93.05 (-2.50)	6324 (-11.1%)	**0.70** (-0.15)
	Reflection Score-based	86.35 (-1.00)	4934 (-13.9%)	1.60 (-1.40)	91.40 (-0.50)	4137 (-1.8%)	0.90 (-0.15)	**95.70** (+0.15)	6896 (-3.0%)	**0.70** (-0.15)
	Probability-based	87.20 (-0.15)	5207 (-9.1%)	2.15 (-0.85)	92.35 (+0.45)	3963 (-6.0%)	0.85 (-0.20)	95.35 (-0.20)	6969 (-2.0%)	0.75 (-0.10)
	Difficulty-informed (ours)	**88.05** (+0.70)	5143 (-10.2%)	2.05 (-0.95)	**92.80** (+0.90)	3990 (-5.3%)	0.65 (-0.40)	95.65 (+0.10)	**6312** (-11.2%)	**0.70** (-0.15)
AIME24	Baseline	48.75	14440	9.58	52.92	13613	6.67	75.42	20363	18.75
	Keyword-based	40.62 (-8.13)	**12658** (-12.3%)	9.58 (+0.00)	44.38 (-8.54)	**12163** (-10.7%)	**5.80** (-0.87)	71.42 (-4.00)	**18774** (-7.8%)	**11.90** (-6.85)
	Reflection Score-based	43.33 (-5.42)	13479 (-6.7%)	**7.92** (-1.66)	51.25 (-1.67)	12914 (-5.1%)	6.67 (+0.00)	73.33 (-2.09)	19796 (-2.8%)	18.33 (-0.42)
	Probability-based	50.63 (+1.88)	13958 (-3.3%)	8.75 (-0.83)	56.67 (+3.75)	12886 (-5.3%)	6.67 (+0.00)	75.21 (-0.21)	19688 (-3.3%)	17.71 (-1.04)
	Difficulty-informed (ours)	**52.50** (+3.75)	14445 (+0.0%)	9.58 (+0.00)	**57.29** (+4.37)	13142 (-3.5%)	6.67 (+0.00)	**77.92** (+2.50)	19254 (-5.4%)	14.17 (-4.58)
GPQA	Baseline	47.79	9895	6.50	46.53	9227	2.08	65.15	9926	**0.00**
	Keyword-based	45.58 (-2.21)	**7303** (-26.2%)	**3.41** (-3.09)	44.51 (-2.02)	**7512** (-18.6%)	**1.70** (-0.38)	62.50 (-2.65)	**8145** (-17.9%)	**0.00** (+0.00)
	Reflection Score-based	46.84 (-0.95)	8827 (-10.8%)	4.04 (-2.46)	46.09 (-0.44)	8987 (-2.6%)	2.02 (-0.06)	64.65 (-0.50)	9391 (-5.4%)	0.13 (+0.13)
	Probability-based	47.98 (+0.19)	9364 (-5.4%)	4.61 (-1.89)	47.41 (+0.88)	9045 (-2.0%)	2.08 (+0.00)	65.15 (+0.00)	9530 (-4.0%)	**0.00** (+0.00)
	Difficulty-informed (ours)	**50.32** (+2.53)	9516 (-3.8%)	4.73 (-1.77)	**49.94** (+3.41)	9019 (-2.3%)	2.02 (-0.06)	**67.11** (+1.96)	9339 (-5.9%)	**0.00** (+0.00)
Average Diff	Keyword-based	-4.36	**-21.9%**	-1.58	-4.52	**-14.3%**	**-0.65**	-3.05	**-12.3%**	**-2.33**
	Reflection Score-based	-2.46	-10.5%	**-1.84**	-0.87	-3.2%	-0.07	-0.81	-3.7%	-0.15
	Probability-based	+0.64	-5.9%	-1.19	+1.69	-4.4%	-0.07	-0.14	-3.1%	-0.38
	Difficulty-informed (ours)	**+2.33**	-4.7%	-0.91	**+2.89**	-3.7%	-0.15	**+1.52**	-7.5%	-1.58

Table 2: Performance comparison of our difficulty-informed reflection steering, keyword-based feature steering, reflection score-based steering, probability-based intervention (Liu et al., 2025), and the baseline. We report the pass@1 accuracy, average token consumption, and cutoff rate. The best result is in **bold**, while the second best is underlined. The change amount compared with the baseline is marked in parentheses. The Average Diff section reports the mean of these variations across the three datasets. Our approach consistently outperforms the baseline and other steering approaches. This advantage is statistically reliable rather than incidental: treating each sample index as an independent run, the gain over the baseline is significant in six of the nine model-dataset configurations under a one-sided paired t-test, and Fisher's combined test over all nine gives $p < 10^{-6}$ for both accuracy and token reduction. Appendix J reports the per-configuration means, standard deviations, and p-values.

input is. Our feature selection is therefore difficulty-informed, while the intervention itself stays static rather than adapting to each input.

Level	Acc. (%)↑	Tokens↓	Cutoff (%)↓
1	93.02 (-0.58)	2605 (-25.7%)	0.00 (-0.58)
2	91.67 (-0.55)	2709 (-16.5%)	0.56 (-0.55)
3	95.00 (0.00)	4295 (-7.8%)	2.14 (+0.47)
4	86.72 (+0.20)	5521 (-3.6%)	2.54 (-1.17)
5	79.85 (+2.98)	7897 (-11.8%)	3.17 (-2.24)

Table 3: Steering performance by Level from the R1-Distill-LLaMA3 model on MATH500 dataset. The change amount compared to the baseline is marked in parentheses. It can be observed that our difficulty-informed steering mainly reduces the token usage for easier problems, while improving accuracy for harder problems.

4.2 Experimental Results

4.2.1 Implementation Details

We evaluate our difficulty-informed reflection steering on three distinct LRMs using their respective Sparse Autoencoders (SAEs): DeepSeek-R1-Distill-LLaMA3-8B (Guo et al., 2025), DeepSeek-R1-Distill-Qwen2.5-7B (Guo et al., 2025), and Qwen3-4B-Thinking-2507 (Team, 2025). For each model, we calculate the steering vector using its own generated responses on the MATH500 dataset. To assess cross-domain generalization, this single MATH500-derived vector is uniformly applied during inference across three challenging reasoning datasets: MATH500 (Lightman et al., 2023), AIME24 (Zhang and Math-AI, 2024), and GPQA DIAMOND (Rein et al., 2024). During evaluation, we sample 4, 16, and 8 solutions per problem for the three datasets, respectively, with a maximum output length of 32,768 tokens. For our approach, we empirically set $\alpha = 5$ across all experiments.

We compare our method against three baselines:

- **Keyword-based feature steering:** A vanilla single-SAE-feature intervention using the feature most activated by the token "wait" (the primary reflection keyword). We scale its L2 Norm to 2 to approximately match our difficulty-informed vector.
- **Reflection score-based steering:** A simplified ablation of our method that ignores the Difficulty Slope and directly uses the Reflection Score as the feature weight, with the same setting of $\alpha = 5$.
- **Probability-based intervention:** ((Liu et al., 2025)) An existing simple but effective baseline that dynamically suppresses the token "wait" whenever its predicted probability falls below a threshold of 0.3.

Steering Setting & Comparisons: Crucially, our approach operates exclusively during standard inference by adding a *fixed* steering vector. It is LLM training-free and remains static regardless of the input query state. This lightweight, "plug-and-play" design ensures minimal computational overhead and seamless integration into existing pipelines. Because of this nature, we restrict our direct comparisons to similarly simple interventions that follow the exact same setting. While other related works exist, many of them require LLM fine-tuning or utilize different steering settings, precluding a fair direct comparison. A detailed discussion of some of these related approaches is provided in the Appendix A.3.

Hyper-parameter α	Acc. (%)↑	Tokens↓	Cutoff (%)↓
0	87.35	5730	3.00
2.5	87.55 (+0.20)	5077 (-11.4%)	2.20 (-0.80)
5	88.05 (+0.70)	5143 (-10.2%)	2.05 (-0.95)
10	87.95 (+0.60)	4995 (-12.8%)	1.85 (-1.15)
20	86.90 (-0.45)	5078 (-11.4%)	2.05 (-0.95)

Table 4: Ablation Study on hyper-parameter α in equation 6. The reduction on token consumption is stable from $\alpha = 2.5$ to 20, while the accuracy reach the highest with $\alpha = 5$.

4.2.2 Comprehensive Comparison

We compared our difficulty-informed approach across three datasets and three LRMs with three baselines: keyword-based and reflection score-based steering, as well as probability-based intervention. (Table 2).

Keyword-based steering maximally reduces token consumption but indiscriminately suppresses reflection, severely degrading accuracy and harming the models' core reasoning capabilities. Reflection score-based steering mitigates this loss but ultimately remains an accuracy-token trade-off. As a strong baseline, the prior probability-based intervention approach (Liu et al., 2025) reduces tokens without sacrificing performance. Notably, our difficulty-informed steering achieves comparable token efficiency while consistently boosting accuracy. Across all models and datasets, our method yields an overall average accuracy improvement of +2.25 and reduces token consumption by 5.3%.

Additionally, our approach demonstrates better stability and generalization on the newer Qwen3-4B-Thinking-2507 model. Because the prior probability-based method relies on rigid keyword probabilities (the "wait" token), its effectiveness diminishes on newer models with probably shifted reasoning distributions. Conversely, our method captures universal difficulty-aware reflection concepts, maintaining robust performance with both accuracy gain and token consumption reduction on the new model.

Finally, although the steering vector is fixed and identical for every query, its effect differs across datasets: token reduction dominates on the easier MATH500, while accuracy gains dominate on the more challenging AIME24 and GPQA. This difference comes from how the model itself responds to the suppressed features, not from any per-input difficulty estimation, and we examine it further in the ablation study below.

4.2.3 Effects on Different Difficulty Levels

We separately compared the steering effect of our approach by level in Table 3. It can be observed that on easier problems, our difficulty-informed steering mainly provides a reduction on the token consumptions. This is reasonable as we suppressed the superficial reflection features mainly activating on easy problems, resulting in less reflection for easier problems. While for harder problems, our steering mainly provides an improvement in accuracy. We conjecture that this is because when the superficial reflection features are suppressed, the model tend to adopt more reflection features that mainly activates on hard problems as compensation, and thus benefits its deep thinking and exploration.

4.2.4 Ablation Study on Hyper-parameters

We conducted an ablation study regarding the hyper-parameter α in equation 6. For α from 2.5 to 20, the effect of token reduction seems stable, while accuracy could also reduce if setting a large α. The best accuracy is achieved with alpha ranging from 5 to 10, and we finally choose $\alpha = 5$ as our universal setting.

Dataset	Baseline	Suppress Easy (ours)	Amplify Hard	Both
MATH500	87.35 / 5730	**88.05 / 5143**	87.90 / 5763	87.75 / 5482
AIME24	48.75 / 14440	**52.50** / 14445	47.92 / 15030	49.17 / 14693
GPQA	47.79 / 9895	**50.32 / 9516**	48.04 / 9848	48.48 / 9921
Avg Δ	–	**+2.33 / -4.7%**	+0.0 / +1.4%	+0.50 / -0.8%

Table 5: Ablation on the steering direction (R1-Distill-LLaMA3, $\alpha = 5$). Each cell reports accuracy (%) / average tokens, and **bold** marks the entries where our suppression variant is the best across all four columns. "Suppress Easy" ($S_i > 0, k_i < 0$) is ours; "Amplify Hard" selects $S_i > 0, k_i > 0$ and adds them; "Both" applies the two jointly.

4.2.5 Ablation Study on Steering Direction

Our method suppresses reflection features biased towards simple problems. A natural alternative is the opposite direction: amplifying the features biased towards hard problems ($S_i > 0, k_i > 0$), or applying both simultaneously. Table 5 compares the three variants on R1-Distill-LLaMA3 at the same $\alpha = 5$. Amplifying hard-problem features brings essentially no accuracy gain while slightly increasing token consumption, and combining both directions recovers only a fraction of the gain obtained by suppression alone. We hypothesize that this asymmetry arises because these LRMs already reflect abundantly: pushing reflection even harder may compete with the actual reasoning steps for the token budget, whereas removing superficial simple-problem reflection frees capacity that the model reallocates on its own.

4.2.6 Cross-Model Generalization

To investigate whether our steering vectors capture universal reflection semantics rather than model-specific artifacts, we evaluated a cross-model transfer setting. Specifically, we calculate the steering vector solely from data generated by R1-Distill-LLaMA3-8B model for the other two LRMs. Results (Table 15 in Appendix) demonstrate transferability: the cross-model-derived vectors consistently outperform the baseline, suggesting that shared underlying reflection concepts exists. Predictably, this transfer incurs a performance discount compared to in-domain steering (using self-generated data), which maximizes performance.

4.2.7 Interpretation of the Steering

We also conducted a semantic analysis of the steered SAE features with the help of LLMs. The results indicate that our method promotes complex reasoning by suppressing superficial self-affirmation and verification processes. Detailed explanations of the analysis method and feature semantics are available in Appendix L. To provide a concrete illustration of the actual steering effect, we present a qualitative case study in Appendix M, demonstrating how our method successfully prevents the model from engaging in redundant, formalistic reflections.

5 Conclusion

In this work, we presented a comprehensive mechanistic interpretability framework to analyze and optimize the reflection capabilities of Large Reasoning Models. Moving beyond brittle, keyword-based heuristics, our automated detection pipeline and the introduction of the Reflection Score provided a robust, unbiased lens into the correlation between SAE features and reflection. By scaling this analysis macroscopically, we demonstrated that LRMs organize reflection concepts geometrically into continuous regions within the hidden state space, functioning collaboratively rather than in isolation. Our analysis further revealed that these reflection regions are structured and difficulty-aware. By introducing the Difficulty Slope metric, we mapped how the model shifts its reflection representations from simple to hard problems. Leveraging this geometric understanding, we designed a difficulty-informed steering intervention to suppress redundant, simple-problem oriented reflection regions during inference. This targeted manipulation successfully prevented LRMs from falling into premature verification loops, consistently enhancing accuracy on complex reasoning tasks while reducing overall token consumption across multiple model architectures and benchmarks. Finally, this paper bridges the gap between microscopic and macroscopic SAE feature analysis and optimization, providing a scalable framework for further research. We hope our framework serves as a foundational step for future research in the field of LRM interpretability.

6 Limitations

Despite the contributions and insights, this study still has several limitations that we plan to explore in the future.

Causality and Theoretical Proof While our framework provides a comprehensive microscopic and macroscopic view of LRM reflection, our SAE-based analysis predominantly relies on correlational observations rather than strict causal mappings. Although the final steering experiments offer some causal evidence by demonstrating performance changes following targeted interventions, a rigorous causal tracing of individual reflection features remains underexplored. Furthermore, several algorithmic design choices and hyperparameter settings (such as the steering intensity α) in our methodology are currently determined empirically. Future works may aim to establish a more robust theoretical foundation and formal mathematical proofs for these geometric representations and steering dynamics.

Static Intervention and Coarse Difficulty Supervision Our intervention is static: difficulty information is used offline to select which features to suppress, and the resulting vector is then added identically to every query without estimating the difficulty of the current input. Estimating the difficulty of each query at inference time and steering accordingly may yield larger gains, at the cost of higher computational and implementation overhead. The Difficulty Slope behind this selection is also fitted on a single dataset (MATH500) using only five coarse annotated levels. Consequently, individual per-feature slopes are relatively noisy, while the aggregated steering vector is more robust.

Dependence on Annotated Reflection Our pipeline defines reflection at the text level through an auxiliary LRM annotator. Reflection, however, has no rigorous formal definition and is ultimately grounded in human judgment, so some annotator variation is unavoidable. Although alternative annotators from different model families agree with ours on roughly 87% to 88% of steps, the agreement is not perfect: the remaining disagreement is partly explained by borderline steps that mix reflection with ordinary reasoning, where different annotators place the threshold differently. A small amount of human calibration or majority voting over several annotators would further improve label reliability. More broadly, deriving such high-level concepts directly from model representations, rather than from text-level labels, remains an open challenge for interpretability research.

Limited Models and Datasets Our experiments were constrained to LRMs with fewer than 8 billion parameters. Scaling our framework to validate these findings on larger, state-of-the-art frontier models remains a critical next step. Finally, our reliance on specific mechanistic interpretability libraries, such as TransformerLens, restricted our ability to evaluate the absolute newest LRM architectures that lack immediate library support. Expanding the compatibility of our framework to broader model architectures is an important direction for future investigation.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. U25A20539.

References

Abien Fred Agarap. 2019. Deep learning using rectified linear units (relu). *Preprint*, arXiv:1803.08375.

Ruichu Cai, Haopeng Du, Qingwen Lin, Yutong Chen, Zijian Li, and Boyan Xu. 2026. Entra: Entropy-based redundancy avoidance in large language model reasoning. *Preprint*, arXiv:2601.07123.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *Preprint*, arXiv:2506.14758.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *Preprint*, arXiv:2309.08600.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. 2024. Transcoders find interpretable LLM feature circuits. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Preprint*, arXiv:2209.10652.

Yi Fang, Wenjie Wang, Mingfeng Xue, Boyi Deng, Fengli Xu, Dayiheng Liu, and Fuli Feng. 2026. Controllable llm reasoning via sparse autoencoder-based steering. *Preprint*, arXiv:2601.03595.

Andrey Galichin, Alexey Dontsov, Polina Druzhinina, Anton Razzhigaev, Oleg Y. Rogov, Elena Tutubalina, and Ivan Oseledets. 2025. I have covered all the bases here: Interpreting reasoning features in large language models via sparse autoencoders. *Preprint*, arXiv:2503.18878.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Zhengfu He, Wentao Shu, Xuyang Ge, Lingjie Chen, Junxuan Wang, Yunhua Zhou, Frances Liu, Qipeng Guo, Xuanjing Huang, Zuxuan Wu, Yu-Gang Jiang, and Xipeng Qiu. 2024. Llama scope: Extracting millions of features from llama-3.1-8b with sparse autoencoders. *Preprint*, arXiv:2410.20526.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. 2024. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440.

Zeju Li, Jianyuan Zhong, Ziyang Zheng, Xiangyu Wen, Zhijian Xu, Yingying Cheng, Fan Zhang, and Qiang Xu. 2026. Making slow thinking faster: Compressing LLM chain-of-thought via step entropy. In *The Fourteenth International Conference on Learning Representations*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Johnny Lin. 2023. Neuronpedia: Interactive reference and tooling for analyzing neural networks. Software available from neuronpedia.org.

Kaiyuan Liu, Chen Shen, Zhanwei Zhang, Junjie Liu, Xiaosong Yuan, and Jieping ye. 2025. Efficient reasoning through suppression of self-affirmation reflections in large reasoning models. *Preprint*, arXiv:2506.12353.

Jian Mu, Qixin Zhang, Zhiyong Wang, Menglin Yang, Shuang Qiu, Chengwei Qin, Zhongxiang Dai, and Yao Shu. 2025. Self-reflective generation at test time. *Preprint*, arXiv:2510.02919.

OpenAI, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, and 107 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *Preprint*, arXiv:2508.10925.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. 2025. Automatically interpreting millions of features in large language models. *Preprint*, arXiv:2410.13928.

Chen Qian, Dongrui Liu, Haochen Wen, Zhen Bai, Yong Liu, and Jing Shao. 2025. Demystifying reasoning dynamics with mutual information: Thinking tokens are information peaks in llm reasoning. *Preprint*, arXiv:2506.02867.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, Janos Kramar, and Neel Nanda. 2025. Jumping ahead: Improving reconstruction fidelity with jumpreLU sparse autoencoders.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Dong Shu, Xuansheng Wu, Haiyan Zhao, Daking Rai, Ziyu Yao, Ninghao Liu, and Mengnan Du. 2025. A survey on sparse autoencoders: Interpreting the internal mechanisms of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 1690–1712, Suzhou, China. Association for Computational Linguistics.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

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

Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. 2025. Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency. *Preprint*, arXiv:2506.08343.

Fengli Xu, Qianyue Hao, Zefang Zong, Jingwei Wang, Yunke Zhang, Jingyi Wang, Xiaochong Lan, Jiahui Gong, Tianjian Ouyang, Fanjin Meng, Chenyang Shao, Yuwei Yan, Qinglong Yang, Yiwen Song, Sijian Ren, Xinyuan Hu, Yu Li, Jie Feng, Chen Gao, and Yong Li. 2025. Towards large reasoning models: A survey of reinforced reasoning with large language models. *Preprint*, arXiv:2501.09686.

Edward Yeo, Yuxuan Tong, Xinyao Niu, Graham Neubig, and Xiang Yue. 2025. Demystifying long chain-of-thought reasoning in LLMs. In *ICLR 2025 Workshop on Navigating and Addressing Data Problems for Foundation Models*.

Kaiyan Zhang, Yuxin Zuo, Bingxiang He, Youbang Sun, Runze Liu, Che Jiang, Yuchen Fan, Kai Tian, Guoli Jia, Pengfei Li, Yu Fu, Xingtai Lv, Yuchen Zhang, Sihang Zeng, Shang Qu, Haozhan Li, Shijie Wang, Yuru Wang, Xinwei Long, and 20 others. 2025a. A survey of reinforcement learning for large reasoning models. *Preprint*, arXiv:2509.08827.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024.

Zhenyu Zhang, Shujian Zhang, John Lambert, Wenxuan Zhou, Zhangyang Wang, Mingqing Chen, Andrew Hard, Rajiv Mathews, and Lun Wang. 2025b. Fantastic reasoning behaviors and where to find them: Unsupervised discovery of the reasoning process. *Preprint*, arXiv:2512.23988.

A Related Works

A.1 Mechanistic Interpretability and Sparse Auto-Encoders

Mechanistic interpretability aims to reverse-engineer the opaque internal workings of neural networks into human-understandable components. A fundamental challenge in this field is polysemanticity, which arises from the superposition hypothesis (Elhage et al., 2022)—the tendency of neural networks to represent more features than they have dimensions by embedding them in almost-orthogonal directions. To disentangle these representations, Sparse Auto-Encoders (SAEs) (Cunningham et al., 2023; Shu et al., 2025) and Transcoders (Dunefsky et al., 2024) have emerged as leading methodologies. By projecting dense hidden states into a higher-dimensional, sparse latent space, SAEs successfully isolate monosemantic, human-interpretable features.

Recent advancements have focused on scaling SAEs to state-of-the-art large language models (Templeton et al., 2024; He et al., 2024; Gao et al., 2025) and developing automated pipelines to interpret the massive number of extracted features (Paulo et al., 2025). While these studies provide a powerful microscopic lens for feature analysis, they predominantly treat SAE features as isolated, discrete vectors. In contrast, our work bridges the gap between microscopic feature extraction and macroscopic structural analysis. By utilizing SAE features as a coordinate basis, we investigate the continuous geometric distribution and clustering of concepts (specifically, reflection) within the LRM's hidden state space.

A.2 Reflection and Reasoning Dynamics in LRMs

The advent of Large Reasoning Models (LRMs) (Xu et al., 2025; Zhang et al., 2025a) has significantly advanced the frontier of complex problem-solving. While earlier generations of Large Language Models notoriously struggled with intrinsic self-correction and self-verification (Huang et al., 2024; Kamoi et al., 2024), modern LRMs exhibit emergent reflection and backtracking capabilities, largely driven by large-scale Reinforcement Learning (Guo et al., 2025). To understand these reasoning dynamics, recent macroscopic studies have explored the training dynamics (Yeo et al., 2025) and utilized metrics like step entropy to compress or analyze Chain-of-Thought (CoT) trajectories (Cai et al., 2026; Li et al., 2026; Cheng et al., 2025). Additionally, test-time generation strategies have been proposed to further elicit self-reflective behaviors (Mu et al., 2025).

From a mechanistic interpretability perspective, researchers have recently begun to probe the internal circuits responsible for reasoning (Zhang et al., 2025b; Galichin et al., 2025). However, current feature-level analyses often rely on heuristic keywords or specific "thinking tokens" (e.g., "wait") to anchor their investigations (Qian et al., 2025; Wang et al., 2025). Furthermore, recent studies have demonstrated that suppressing superficial self-affirmation or specific thinking tokens can streamline reasoning without degrading performance (Liu et al., 2025; Wang et al., 2025). Our research advances this paradigm in two crucial ways. First, we replace brittle keyword heuristics with a robust, LRM-annotated detection framework that calculates a comprehensive "Reflection Score." Second, we uncover an intrinsic, difficulty-aware structure within the reflection latent space, enabling a more nuanced, targeted steering strategy that selectively suppresses simple-problem reflection rather than unconditionally ablating specific tokens.

A.3 Methodological Differences with Related Works

While some recent studies also explore LLM reasoning control, fundamental differences in experimental settings and training requirements preclude direct comparisons with our steering approach:

- **Inference Overhead:** (Zhang et al., 2025b) uses test-time training to dynamically optimize sample-specific hyperparameters, introducing substantial computational overhead. In contrast, our universal static vector incurs negligible test-time latency.

- **Error-Correction vs. Standard Generation:** (Fang et al., 2026) evaluates error recovery after erroneous trajectories. Conversely, our method targets standard, from-scratch reasoning generation without assuming pre-existing flawed contexts.

- **Vector Search vs. Direct Computation:** While (Galichin et al., 2025) also applies a universal inference steering vector, they rely on a costly brute-force search on validation sets. We improve efficiency by computing

the vector directly from SAE features using designed metrics.

- **Training Requirements:** Methods such as (Cai et al., 2026) and (Cheng et al., 2025) require modifying LLM weights via supervised fine-tuning or Reinforcement Learning (RL). Our approach is LLM training-free, operating purely as a plug-and-play intervention during standard inference.

B Detailed Formulation of Sparse Auto-Encoders

B.1 Background: Polysemanticity and the Superposition Hypothesis

A fundamental challenge in the mechanistic interpretability of Large Language Models (LLMs) is the phenomenon of polysemanticity. According to the "superposition hypothesis" (Elhage et al., 2022), neural networks tend to represent more features than they have dimensions by embedding them as almost-orthogonal vectors. Consequently, individual neurons often activate in response to multiple unrelated concepts, rendering the dense hidden states highly entangled and difficult for humans to interpret.

To untangle these dense representations, Sparse Auto-Encoders (SAEs) has emerged as a standard technique (Cunningham et al., 2023). The core objective is to reconstruct these sparse, almost-orthogonal feature vectors through unsupervised learning.

B.2 Network Structure, Input, and Output

An SAE is an Auto-Encoder network that reconstruct inputs by projecting them into a higher-dimensional, sparse latent space.

Given a dense activation vector $\mathbf{x} \in \mathbb{R}^{d_{model}}$, the encoder first computes a sparse intermediate feature vector $\mathbf{f} \in \mathbb{R}^{d_{SAE}}$ (where $d_{SAE} \gg d_{model}$) via a linear projection and a non-linear activation function:

$$\mathbf{f} = \delta(\mathbf{W}_{enc}\mathbf{x} + \mathbf{b}_{enc}) \tag{7}$$

where $\mathbf{W}_{enc} \in \mathbb{R}^{d_{SAE} \times d_{model}}$ is the encoder weight matrix, and $\mathbf{b}_{enc} \in \mathbb{R}^{d_{SAE}}$ is the encoder bias. The activation function δ, typically a ReLU (Agarap, 2019), ensures that all features activate non-negatively.

The decoder then reconstructs the original dense activation vector $\mathbf{x}$ from $\mathbf{f}$. Using a dictionary weight matrix $\mathbf{W}_{dec} \in \mathbb{R}^{d_{model} \times d_{SAE}}$, the reconstructed output $\hat{\mathbf{x}} \in \mathbb{R}^{d_{model}}$ is calculated as:

$$\hat{\mathbf{x}} = \mathbf{W}_{dec}\mathbf{f} \tag{8}$$

Consequently, the original activation vector can be decomposed into a weighted sum of the decoder matrix columns and a reconstruction error term $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$:

$$\mathbf{x} = \sum_{i=1}^{d_{SAE}} \mathbf{f}_i ||\mathbf{W}^i_{dec}|| \cdot \frac{\mathbf{W}^i_{dec}}{||\mathbf{W}^i_{dec}||} + \mathbf{e} \tag{9}$$

Under the linear representation hypothesis, the normalized columns of the decoder matrix $\mathbf{W}^i_{dec}/||\mathbf{W}^i_{dec}||$ serve as "feature directions" aligned with meaningful, monosemantic concepts. The scaled activation value $\mathbf{f}_i||\mathbf{W}^i_{dec}||$ denotes the intensity of that specific concept. By enforcing sparsity on the intermediate vector $\mathbf{f}$, any dense activation vector $\mathbf{x}$ can thus be decomposed into a sparse linear combination of interpretable concepts.

B.3 Objective Function and Interpret Feature Vectors

To guarantee that the learned features remain faithful to the original input while maintaining sparse activation, the SAE optimizes a trade-off between reconstruction accuracy and latent sparsity. The standard loss function $\mathcal{L}$ combines a Mean Squared Error (MSE) reconstruction loss with an L_1 regularization penalty on the feature activation intensity:

$$\mathcal{L} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \lambda\|\mathbf{f}\|_1$$

where λ is a hyperparameter controlling the sparsity penalty. Alternative SAE architectures employ different sparsity mechanisms, such as Top-K (Gao et al., 2025) and JumpReLU (Rajamanoharan et al., 2025), to further improve reconstruction fidelity and sparsity.

After training, individual feature vectors $\mathbf{W}^i_{dec}/||\mathbf{W}^i_{dec}||$ can be interpreted by analyzing their activating contexts. For example, a feature consistently activating on words like "yes," "true," and "positive" likely encodes the concept of affirmation. Auto-Interpretation approaches utilizing LLMs are also widely adopted to scale the semantic summarization of these features (Paulo et al., 2025; Gao et al., 2025).

In Table 6, we summarized the SAEs used in this paper.

LRM	Input Activation	Input Dimension	Intermediate Dimension	Source
DeepSeek-R1-Distill-LLaMA3-8B	15.residual-post	4096	32768	Pretrained, open source
DeepSeek-R1-Distill-Qwen2.5-7B	14.residual-post	3584	57344	Trained by us
Qwen3-4B-Thinking-2507	17.residual-post	2560	40960	Trained by us

Table 6: An overview of the SAEs used in this paper. The input activation is in the format of "layer.position", e.g., 15.residual-post means the hidden state of the residual stream after transformer layer 15.

C Step Classification Prompt

Below is the prompt we used for classifying steps into Reflection Step or Reasoning Step.

> User: Please identify if the following sentence belongs to "Reflection Step" or "Common Reasoning Step", and put your final answer of either A:"Reflection Step" or B:"Common Reasoning Step" within \boxed{}.
>
> <step>
>
> Assistant:

D Reflection Score Analysis

We sampled several SAE features with top-100 Reflection Scores, and manually inspected their activation contexts to summarize an explanation, as recorded in Table 8. The top reflection features exhibit a variety of reflection related concepts.

E Cross-Annotator Validation

Our reflection labels come from a single auxiliary LRM (DeepSeek-R1-Distill-Qwen-14B), so we verify that the labels do not depend on that particular choice. We randomly sampled 50 MATH500 problems (10 per difficulty level) and drew 2,000 steps from their reasoning trajectories, then reclassified every step independently with two additional annotators from different model families using the same prompt as in Appendix C: DeepSeek-V4-Pro (a thinking model) and Qwen3.7-Plus (a non-thinking model).

As shown in Table 9, both alternative annotators agree with the original one on 87% to 88% of steps. The residual disagreement is systematic rather than random: both alternative annotators label reflection somewhat more liberally (21.2% and 17.8% versus 15.6%). Part of this gap comes from borderline steps that interleave reflection with ordinary reasoning, where the threshold is genuinely a matter of judgment.

F Preliminary Experiments regarding Difficulty

We conduct our preliminary experiments regarding difficulty on the R1-Distill-LLaMA3 model. We utilize the difficulty level annotation and independently compute the Reflection Scores for each level using the methodology described in Section 2. We again use UMAP to visualize the distribution of Reflection Scores by level in Figure 3. We observed a slight transition of the orange color from the bottom left to the upper right part, indicating that the center of the reflection region shifts towards the upper right as the difficulty increases, which is further confirmed in Figure 2c.

G Robustness of the Geometric Analysis

Shuffled-feature null control. The neighborhood correlations reported in Section 3 could in principle arise from the geometry of the SAE decoder alone. To rule this out, we randomly permute the Reflection Scores (respectively the Difficulty Slopes) across features while keeping the decoder matrix untouched, and recompute the neighborhood correlation with the same $k = \sqrt{d_{SAE}}$. Under this null, any correlation that survives can only come from the decoder geometry, because the association between a feature direction and its score has been destroyed.

Table 10 shows that the shuffled correlations collapse to zero within noise for both metrics and all three models, two orders of magnitude below the observed values. The continuity we report is therefore a property of how the model assigns reflection semantics to directions, not an artifact of the decoder geometry.

Feature counts and steering-vector sensitivity. Table 11 reports how many features the filter in Equation 5 selects, together with the cosine similarity between the full steering vector and vectors rebuilt from only the top-$K\%$ features by $|w_i|$.

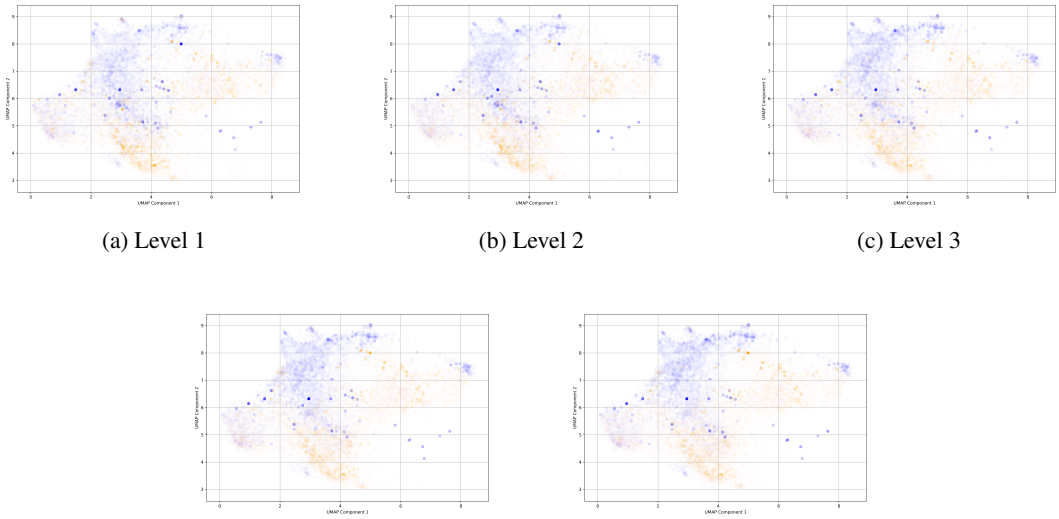
The steering vector is strongly concentrated: for every model, the top 1% of selected features (53 to 67 features) already reproduces the full vector

Keyword	Prob. RS Start With	Prob. following RS	Reflection Step Example
Wait	**44.7%**	**54.2%**	Wait, that's still negative. Hmm, maybe I need a better method.
But	13.2%	29.1%	But BC is (1,-4), AD is (3,6). These are not equal.
Alternatively	10.5%	48.7%	Alternatively, let's verify the cycle again:
So	6.8%	9.1%	So, going back, n(x) is even.
Therefore	2.9%	9.5%	Therefore, 2022 is out. So, the four-digit numbers don't work either.
Hmm	2.6%	41.1%	Hmm, this is getting complicated. Maybe another approach is better.

Table 7: The top-6 first token of reflection steps and their statistics. "Prob. RS Start With" is the probability a Reflection Step starts with that keyword, while "Prob. following RS" is the probability that keyword followed by a Reflection Step.

Rank	Feature ID	Reflection Score	Human Summarized Explanation
1	9523	63.1	Common feature that activates on various texts.
2	16267	44.2	".\n\n" at the end of a step.
3	5008	39.3	The comma after "wait".
4	31172	36.4	The comma after "wait" and "no".
5	18410	35.3	The word "wait".
16	20456	20.5	Expressions about confirmation and checking.
33	13548	12.2	The word "me" in sentences like "let me verify".
43	7760	11.0	Expressions about verification.
59	10274	9.9	Words like "wait", "perhaps", "clarify", etc.
77	9842	8.9	Expressions about detecting mistakes and pursuing alternative solutions.

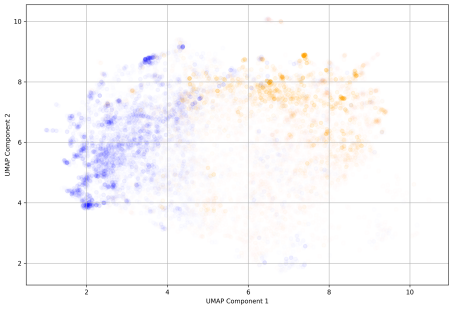
Table 8: An summarization of some sampled features from R1-Distilled-LLaMA3's SAE with top-100 Reflection Scores. We summarize the explanations by their top activations.



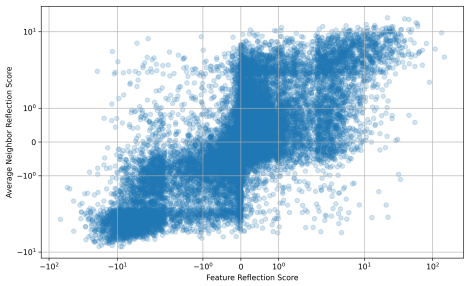
(a) Level 1 (b) Level 2 (c) Level 3

(d) Level 4 (e) Level 5

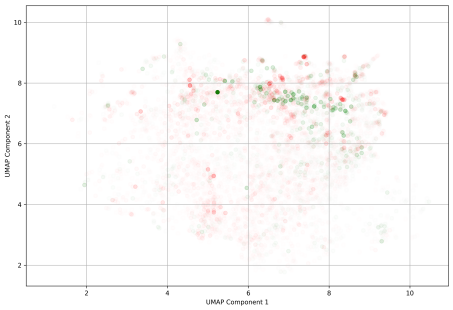
Figure 3: The UMAP Visualization of Reflection Scores calculated by difficulty levels. A slight transition of the orange color from the bottom left to the upper right part is observed.



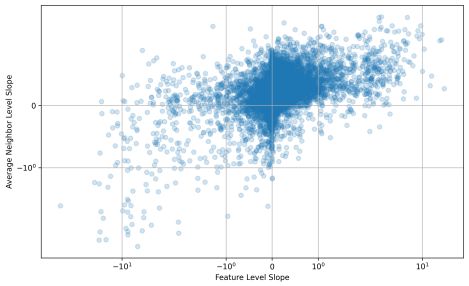
(a) UMAP of Reflection Score



(b) Neighborhood Correlation (RS, $r = 0.586$)

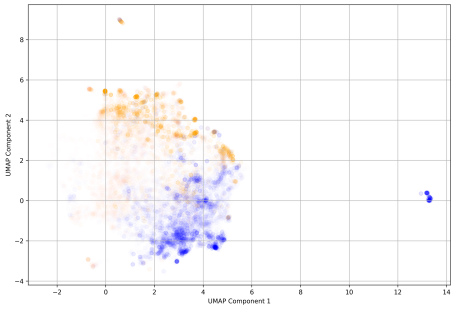


(c) UMAP of Difficulty Slope

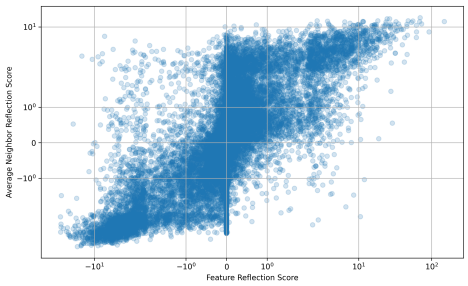


(d) Neighborhood Correlation (DS, $r = 0.429$)

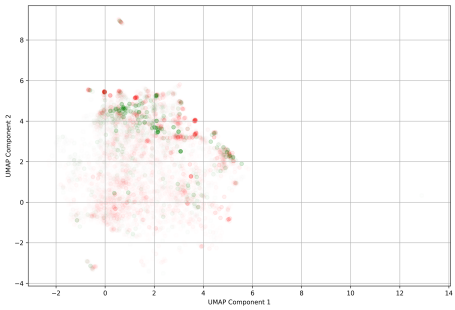
Figure 4: Visual evidence of macroscopic continuous representation and difficulty-aware sub-regions in the hidden state space of DeepSeek-R1-Distill-Qwen2.5-7B.



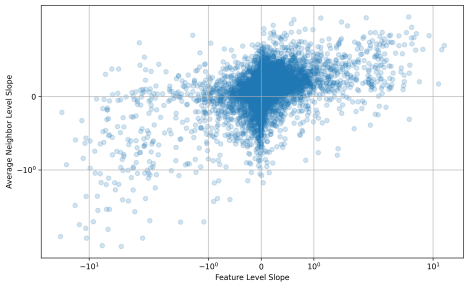
(a) UMAP of Reflection Score



(b) Neighborhood Correlation (RS, $r = 0.630$)



(c) UMAP of Difficulty Slope



(d) Neighborhood Correlation (DS, $r = 0.459$)

Figure 5: Visual evidence of macroscopic continuous representation and difficulty-aware sub-regions in the hidden state space of Qwen3-4B-Thinking-2507.

Annotator	N	Agr. (%)	Refl. (%)
Original	–	–	15.6
Human	400	93.25	–
DeepSeek-V4-Pro	2000	86.8	21.2
Qwen3.7-Plus	2000	88.2	17.8

Table 9: Cross-annotator agreement against the original annotator (DeepSeek-R1-Distill-Qwen-14B). "Agr." is step-level agreement and "Refl." is the fraction of steps each annotator labels as reflection. The human row repeats the manual verification of Table 1, measured on a separate sample of 400 steps.

Model	Reflection Score		Difficulty Slope	
	Original	Shuffled	Original	Shuffled
LLaMA3	0.487	-0.001 ± 0.006	0.268	0.002 ± 0.007
Qwen2.5	0.586	0.001 ± 0.003	0.429	-0.002 ± 0.004
Qwen3	0.630	0.001 ± 0.006	0.459	0.002 ± 0.006

Table 10: Shuffled-feature control for the neighborhood correlations. "Original" repeats the Pearson r reported in Section 3 and Appendix I; "Shuffled" is the mean $\pm$ std over 10 random permutations of the scores across features. All shuffled correlations are indistinguishable from zero.

Model	d_{SAE}	$S_i > 0, k_i < 0$	top-1%	top-5%	top-10%
LLaMA3	32768	6766 (20.6%)	0.985	0.999	1.000
Qwen2.5	57344	5381 (9.4%)	0.995	1.000	1.000
Qwen3	40960	5826 (14.2%)	0.998	1.000	1.000

Table 11: Number of features selected by Equation 5, and the cosine similarity between the full steering vector and vectors built from only the top-$K\%$ features by absolute steering coefficient. Percentages are relative to d_{SAE}.

at a cosine similarity above 0.98, and the top 5% is indistinguishable from it. This matters for the concern that features lying close to the sign boundary of Equation 5 may be misclassified by slope noise, because such features necessarily carry small $|S_i \cdot k_i|$ and thus contribute negligibly to the aggregate direction. Incidentally, this concentration also suggests that the highest-weight features sit near the center of the sub-region rather than at its periphery.

H Robustness of the Difficulty Slope

The Difficulty Slope of each feature is fitted by linear regression on only five annotated difficulty levels, which raises two questions: how reliable is an individual slope, and does the metric survive when the annotated levels are unavailable? We address both on the R1-Distill-LLaMA3 model.

Bootstrap confidence intervals. We resample the 500 MATH500 problems with replacement (1,000 iterations), recompute the Difficulty Slope of every feature for each resample, and compare the refitted values with the original ones. They are stable: the cosine similarity is 0.941 with a 95% interval of $[0.829, 0.985]$.

Individual slopes are nonetheless noisier than their aggregate, which is a trade-off we accept for the simplicity of the five-level fit. This is precisely why our steering aggregates thousands of features into one vector: the vector is a weighted average, so per-feature noise is smoothed out, and Table 11 shows that the aggregate is anchored by the high-magnitude features. Low-weight features near the decision boundary may occasionally have their sign flipped, but they have minimal impact on the aggregated direction.

Label-free difficulty proxies. The Difficulty Slope as defined in Section 3 needs a per-problem difficulty label, which most datasets do not provide. We therefore replace the annotated levels with three model-intrinsic proxies, re-bin the problems into five quantile bins by each proxy, refit all slopes, and correlate them with the original annotation-based slopes: (1) **output length**, the average number of tokens the model generates for the problem; (2) **LRM annotation**, a difficulty rating produced by DeepSeek-V4-Pro given the problem text and the model's reasoning trace; and (3) **solve rate**, the empirical fraction of correct samples.

As Table 12 shows, output length and LRM an-

Difficulty Proxy	Slope Pearson r
Annotated level (original)	1.000
Output length	0.886
LRM annotation	0.930

Table 12: Correlation between the Difficulty Slopes refitted from each label-free proxy and the original annotation-based slopes, on R1-Distill-LLaMA3.

notation both recover the original slopes closely ($r = 0.886$ and 0.930), so either can substitute for human difficulty labels. Output length is the more attractive of the two since it is fully intrinsic and requires no auxiliary model; it increases monotonically with annotated difficulty, from a mean of 1,878 tokens at level 1 to 6,344 at level 5. Solve rate, in contrast, is not usable here: with only 4 samples per problem its granularity is too coarse to be effective.

Binning granularity. Using output length as the proxy, we vary the number of quantile bins. As Table 13 shows, the refitted slopes remain stable across 3, 5, and 10 bins. The five-level granularity of the original annotation is therefore not a critical choice.

Bins	Slope Pearson r
3	0.891
5	0.886
10	0.872

Table 13: Sensitivity of the Difficulty Slope to the number of quantile bins used with the output-length proxy, on R1-Distill-LLaMA3. Correlations are against the original annotation-based slopes.

I Visualizations for Additional Models

In Section 3, we presented the macroscopic and difficulty-aware geometric analysis primarily using the DeepSeek-R1-Distill-LLaMA3-8B model. Here, we provide the corresponding visual evidence for the other two evaluated models: DeepSeek-R1-Distill-Qwen2.5-7B (Figure 4) and Qwen3-4B-Thinking-2507 (Figure 5). These visualizations confirm that both the general continuous representation of reflection and the difficulty-aware sub-regions consistently exist across different LRM architectures.

J Variance and Significance Analysis

The main results in Table 2 report point estimates. Because our evaluation already samples multiple solutions per problem (4, 16, and 8 for MATH500, AIME24, and GPQA respectively), we can treat each sample index as one independent run of the whole benchmark, which yields several accuracy and token estimates per configuration. We report their mean $\pm$ standard deviation, and test the baseline against our method with a one-sided paired t-test, pairing runs by sample index. We then combine the nine per-configuration p-values with Fisher's method.

Table 14 shows that accuracy improves in all nine configurations, with six of the nine gains individually significant, and tokens decrease in eight of nine, with six of those significant. Aggregated by Fisher's method, both effects are highly significant ($p < 10^{-6}$).

The two metrics are strongest on complementary regimes. On MATH500 the accuracy headroom is small because the baselines are already at 87% to 96%, and the benefit shows up as token reduction (-5.3% to -11.2%, significant on all three models). On AIME24 and GPQA the accuracy gains are larger, which is consistent with the level-wise behavior in Table 3. AIME24 shows the widest standard deviations because it contains only 30 problems; we mitigate this by sampling 16 solutions per problem (480 generations in total), but the per-run spread remains large and Qwen3's accuracy gain there does not reach individual significance.

K Cross-model Generalization

To investigate whether our steering vectors capture universal reflection semantics rather than model-specific artifacts, we evaluated a cross-model transfer setting. Specifically, we calculate the steering vector solely from data generated by R1-Distill-LLaMA3-8B model for the other two LRMs. Results (Table 15) demonstrate transferability: the cross-model-derived vectors consistently outperform the baseline, suggesting that shared underlying reflection concepts exists. Predictably, this transfer incurs a performance discount compared to in-domain steering (using self-generated data), which maximizes performance.

L Interpretation of the Steering Vector

An advantage of utilizing SAEs for steering is the inherent interpretability of the features being ma-

Model	Dataset	Accuracy (%) Baseline	Accuracy (%) Ours	Accuracy (%) Δ	Accuracy (%) p	Tokens Baseline	Tokens Ours	Tokens Δ	Tokens p
LLaMA3	MATH500	87.35 ± 0.85	88.05 ± 1.29	+0.70	0.055	5730 ± 386	5143 ± 217	-10.2%	**0.040**
	AIME24	48.75 ± 7.49	52.50 ± 5.64	+3.75	**0.048**	14440 ± 1816	14445 ± 1394	$+0.0\%$	0.504
	GPQA	47.79 ± 1.50	50.32 ± 1.26	+2.53	**0.002**	9895 ± 479	9516 ± 671	-3.8%	0.096
Qwen2.5	MATH500	91.90 ± 0.38	92.80 ± 0.28	+0.90	**0.011**	4215 ± 107	3990 ± 101	-5.3%	**0.043**
	AIME24	52.92 ± 5.00	57.29 ± 7.62	+4.37	**0.025**	13613 ± 766	13142 ± 455	-3.5%	**0.023**
	GPQA	46.53 ± 1.72	49.94 ± 1.44	+3.41	**0.001**	9227 ± 180	9019 ± 290	-2.3%	0.080
Qwen3	MATH500	95.55 ± 0.55	95.65 ± 0.44	+0.10	0.091	7111 ± 230	6312 ± 181	-11.2%	**0.002**
	AIME24	75.42 ± 6.07	77.92 ± 6.54	+2.50	0.181	20363 ± 1261	19254 ± 965	-5.4%	**0.001**
	GPQA	65.15 ± 1.01	67.11 ± 0.99	+1.96	**0.004**	9926 ± 653	9339 ± 612	-5.9%	**0.048**
Fisher combined				+2.25	$< 10^{-6}$			-5.3%	$< 10^{-6}$

Table 14: Variance and significance of the results in Table 2. Each sample index is treated as one independent run, giving 4/16/8 runs per configuration for MATH500/AIME24/GPQA. We report mean $\pm$ standard deviation across runs, and one-sided paired t-test p-values with significant values ($p < 0.05$) in **bold**. The last row aggregates the nine per-configuration p-values with Fisher's combined test; its Δ column reports the unweighted average across configurations.

Setting		MATH500			AIME24			GPQA		
Target Model	**Method**	**Acc. (%)↑**	**Tokens↓**	**Cutoff (%)↓**	**Acc. (%)↑**	**Tokens↓**	**Cutoff (%)↓**	**Acc. (%)↑**	**Tokens↓**	**Cutoff (%)↓**
DeepSeek-R1-Distill-Qwen2.5-7B	Baseline	91.90	4215	1.05	52.92	13613	**6.67**	46.53	9227	2.08
	Cross-Model Steering	92.10 (+0.30)	4088	**0.65**	55.21 (+2.29)	13146	7.50	49.56 (+3.03)	9380	2.65
	In-Domain Steering	**92.80** (+0.90)	**3990**	**0.65**	**57.29** (+4.37)	**13142**	**6.67**	**49.94** (+3.41)	**9019**	**2.02**
Qwen3-4B-Thinking-2507	Baseline	95.55	7111	0.85	75.42	20363	18.75	65.15	9926	**0.00**
	Cross-Model Steering	**95.70** (+0.15)	6673	0.80	77.29 (+1.87)	19363	**14.17**	65.34 (+0.19)	9408	**0.00**
	In-Domain Steering	95.65 (+0.10)	**6312**	**0.70**	**77.92** (+2.50)	**19254**	**14.17**	**67.11** (+1.96)	**9339**	**0.00**

Table 15: Comparison of cross-model generalization versus in-domain steering for our difficulty-informed method. "Cross-Model Steering" relies on data generated from DeepSeek-R1-Distill-LLaMA3-8B to calculate steering vector for Qwen-series models. While cross-model steering consistently outperforms the baseline, indicating generalizability, in-domain steering preserves the maximum performance gain.

nipulated. To understand exactly what cognitive behaviors our steering vector is affecting, we analyzed the semantics of the features with the highest and lowest steering coefficients w_i (we reset the constraints in equation 5 to $S_i > 0$) to analyze the difference between features activating mainly on easy or hard problems.

We select the top and bottom 100 features from the R1-Distill-LLaMA3's SAE and obtained their corresponding explanations from the open Neuronpedia Database[1] (Lin, 2023). We summarized the difference between the two sets of feature explanations with the help of a powerful LLM (Gemini 3 in our case). The prompt we used is as follows:

> Summarize the common and different characteristics of the following two sets of features:
>
> Top 100 features with the highest steering coefficients:
>
> <list of top 100 features' explanations>
>
> Bottom 100 features with the lowest steering coefficients:
>
> <list of bottom 100 features' explanations>

The summarized explanations of the two sets are:

- **Exploratory Phase (Top 100 Features):** These features characterize the messy, exploratory phase of reasoning. They activate on hesitation keywords (e.g., "wait", "but"), raw uncertainty, speculation, and structural building blocks of the problem. They represent the model in the thick of processing and brainstorming.
- **Evaluative Phase (Bottom 100 Features):** Conversely, these features are heavily skewed toward metacognition, verification, and confirmation. They correspond to states of evaluating work, rechecking for errors, making affirmative statements, and confirming conclusions.

By comparing these two sets, it becomes evident that our steering method primarily identifies and suppresses reflection concepts related to verification and confirmation of existing reasoning process. In simple problems, the model frequently utilizes these features for superficial verification and confirmation, "wasting" the output tokens; while on complex tasks, falling into such superficial confirmation and reflection might hinders the model from engaging in truly meaningful exploration. Therefore, we can promote LRM's performance by suppressing these sub-regions in the hidden state space. This finding is consistent with several prior works (Liu et al., 2025) that find suppressing model's self-affirmation reflection can shorten its answer while not severely harming its performance.

[1] https://www.neuronpedia.org/deepseek-r1-distill-llama-8b/15-llamascope-slimpj-openr1-res-32k/

M Qualitative Case Study

We present a qualitative case study from the MATH500 dataset, comparing the reasoning trajectories before (baseline) and after applying our difficulty-informed steering.

Problem: Let

$$x^8 + 3x^4 - 4 = p_1(x)p_2(x)\cdots p_k(x),$$

where each non-constant polynomial $p_i(x)$ is monic with integer coefficients, and cannot be factored further over the integers. Compute $p_1(1) + p_2(1) + \cdots + p_k(1)$.
Ground Truth: 10

Analysis:

Both the unsteered baseline and our steered model correctly initiate the factorization by substituting $y = x^4$, decomposing the polynomial into $(x^4 + 4)(x^4 - 1)$. However, the unsteered baseline prematurely halts the factorization process after expanding $x^4 - 1$ into $(x^2 - 1)(x^2 + 1)$ by hallucinating a false mathematical premise (incorrectly stating their discriminants are negative). Instead of actually correcting the error, the unsteered model generates extensive, keyword-rich reflection (e.g., *"Wait, but let me count the degrees: ..."* and *"Wait, is that all? Let me double-check. Did I miss any factors? ..."*) to superficially check only if the sum of degrees equals 8, ultimately arriving at the incorrect answer of 8.

In contrast, our difficulty-informed steered model efficiently applies the difference of squares to factor $x^4 - 1$ into its irreducible components without redundant reflection steps, yielding the correct sum of 10. The only reflection step is observed after obtaining the final answer, where the model briefly checks its calculations to ensure correctness. This observation highlights how our steering

method successfully suppresses superficial reflections to produce cleaner, more rigorous reasoning trajectories.

Raw Generation Outputs:

```
Assistant: <think>
Okay, so I have this polynomial: \(x^8 + 3x^4 - 4\), and I need to factor it into polynomials with integer coefficients that can't be factored further. Then, after finding all those factors, I need to compute the sum of each of them evaluated at \(x = 1\). Hmm, that sounds a bit tricky, but let's see how I can approach it.

First, I remember that factoring higher-degree polynomials can sometimes be done by recognizing patterns or using substitution. The given polynomial is an eighth-degree polynomial, which is quite high, but maybe it can be factored into smaller-degree polynomials. Let me look at the structure: \(x^8 + 3x^4 - 4\). It seems like the exponents are multiples of 4, except for the constant term. That makes me think of substituting \(y = x^4\), which would transform the equation into a quadratic in terms of \(y\). Let me try that.

Let \(y = x^4\), then the equation becomes:
\[
y^2 + 3y - 4 = 0
\]
Hmm, okay, so this is a quadratic equation. Let me solve for \(y\) using the quadratic formula. The quadratic is \(y^2 + 3y - 4 = 0\), so:
\[
y = \frac{-3 \pm \sqrt{9 + 16}}{2} = \frac{-3 \pm \sqrt{25}}{2} = \frac{-3 \pm 5}{2}
\]
So, the solutions are:
\[
y = \frac{2}{2} = 1 \quad \text{and} \quad y = \frac{-8}{2} = -4
\]
Therefore, \(y = 1\) or \(y = -4\). Remembering that \(y = x^4\), we have two equations:
1. \(x^4 = 1\)
2. \(x^4 = -4\)

Let me solve each of these.

For the first equation, \(x^4 = 1\). Taking the fourth root of both sides, we get \(x = \pm 1\), or \(x = \pm i\). But since we're factoring over integers, complex roots won't help directly here. However, they might suggest that the polynomial can be factored into quadratics or other polynomials with real coefficients.

For the second equation, \(x^4 = -4\). Taking the fourth root of both sides, we get \(x = \pm \sqrt[4]{-4}\). That would involve complex numbers again, but perhaps we can express it in terms of real quadratics.

Wait, perhaps I should think about the original polynomial differently. If it factors into polynomials with integer coefficients, maybe it factors into quadratics or quartic polynomials. Let me consider whether it's a biquadratic or not.

Alternatively, since it's a biquadratic (all exponents are even), maybe it can be factored into two quartic polynomials or four quadratic polynomials. Let me try factoring it as a product of two quadratics first.

Let me suppose that:
\[
x^8 + 3x^4 - 4 = (x^4 + ax^3 + bx^2 + cx + d)(x^4 + ex^3 + fx^2 + gx + h)
\]
But that seems complicated because both sides have degree 8, and expanding that product would be messy. Maybe I should instead try to factor it as a product of two quadratics in terms of \(x^4\). Wait, but it's a quadratic in \(x^4\), so perhaps it can be factored as a product of two quadratics in \(x^2\). Let me try that.

Suppose:
\[
y^2 + 3y - 4 = (y + a)(y + b)
\]
But wait, if I factor \(y^2 + 3y - 4\), I get \((y + 4)(y - 1)\), right? Because \(4 \times (-1) = -4\) and \(4 + (-1) = 3\). So, that's correct. So, substituting back, we have:
\[
x^8 + 3x^4 - 4 = (x^4 + 4)(x^4 - 1)
\]
Wait, but \(x^4 - 1\) can be factored further, right? Because \(x^4 - 1\) is a difference of squares: \(x^4 - 1 = (x^2)^2 - 1^2 = (x^2 - 1)(x^2 + 1)\). Then, \(x^2 - 1\) can be factored further into \((x - 1)(x + 1)\). So, putting it all together:
\[
x^8 + 3x^4 - 4 = (x^4 + 4)(x^2 - 1)(x^2 + 1)
\]
But \(x^2 - 1\) and \(x^2 + 1\) are both quadratics that can't be factored further over the integers, as their
```

```
    discriminants are negative. So, now
    we have three factors: \(x^4 + 4\),
    \(x^2 - 1\), and \(x^2 + 1\). But
    wait, \(x^4 + 4\) can also be
    factored further, right? I recall
    that \(x^4 + 4\) is a sum of squares
    , which can be factored into two
    quadratics. Let me try that.

Yes, \(x^4 + 4\) can be factored as \((x
    ^2 + 2x + 2)(x^2 - 2x + 2)\). Let me
     check that:
\[
(x^2 + 2x + 2)(x^2 - 2x + 2) = x^4 - 2x
    ^3 + 2x^2 + 2x^3 - 4x^2 + 4x + 2x^2
    - 4x + 4
\]
Simplify:
- \(x^4\)
- \(-2x^3 + 2x^3 = 0\)
- \(2x^2 - 4x^2 + 2x^2 = 0\)
- \(4x - 4x = 0\)
- \(+4\)
So, it's \(x^4 + 4\), which is correct.
    So, now we can factor \(x^4 + 4\)
    into two quadratics, each of which
    is irreducible over integers because
     their discriminants are negative:
- For \(x^2 + 2x + 2\), discriminant is
    \(4 - 8 = -4\)
- For \(x^2 - 2x + 2\), discriminant is
    \(4 - 8 = -4\)

So, now, putting it all together, the
    original polynomial factors into:
\[
(x^2 + 2x + 2)(x^2 - 2x + 2)(x^2 - 1)(x
    ^2 + 1)
\]
Wait, but let me count the degrees: each
     quadratic is degree 2, so 2 + 2 + 2
     + 2 = 8, which matches. So, that's
    four factors. Now, the problem
    states that each non-constant
    polynomial \(p_i(x)\) is monic with
    integer coefficients and cannot be
    factored further over integers. So,
    these four quadratics are
    irreducible, so they are the
    complete factorization.

Therefore, the factors are:
1. \(x^2 + 2x + 2\)
2. \(x^2 - 2x + 2\)
3. \(x^2 - 1\)
4. \(x^2 + 1\)

Now, the problem asks to compute the sum
     \(p_1(1) + p_2(1) + p_3(1) + p_4(1)
    \). So, let's compute each \(p_i(1)
    \):

1. For \(x^2 + 2x + 2\), plug in \(x=1\)
    :
\[
1^2 + 2(1) + 2 = 1 + 2 + 2 = 5
\]
2. For \(x^2 - 2x + 2\), plug in \(x=1\)
    :
\[
1^2 - 2(1) + 2 = 1 - 2 + 2 = 1
\]
3. For \(x^2 - 1\), plug in \(x=1\):
\[
1^2 - 1 = 1 - 1 = 0
\]
4. For \(x^2 + 1\), plug in \(x=1\):
\[
1^2 + 1 = 1 + 1 = 2
\]
So, adding them up: 5 + 1 + 0 + 2 = 8.

Wait, is that all? Let me double-check.
    Did I miss any factors? The original
     polynomial was degree 8, and after
    factoring, I have four quadratics,
    each degree 2, which adds up
    correctly. Each quadratic is monic,
    irreducible over integers, so I
    think that's the complete
    factorization.

Alternatively, just to be thorough, let
    me check if \(x^4 + 4\) can be
    factored differently or if there's
    another way to factor the original
    polynomial. But I think the way I
    did it is correct because
    substitution led me to the
    factorization.

So, I think the sum is 5 + 1 + 0 + 2 =
    8. Therefore, the final answer
    should be 8.

**Final Answer**
\boxed{8}
<|reserved_special_token_6|>

To factor the polynomial \(x^8 + 3x^4 -
    4\), we start by recognizing it as a
     quadratic in terms of \(y = x^4\).
    This transforms the equation into:

\[
y^2 + 3y - 4 = 0
\]

Solving this quadratic equation using
    the quadratic formula, we get the
    roots:

\[
y = 1 \quad \text{and} \quad y = -4
\]

Substituting back \(y = x^4\), we have:

1. \(x^4 = 1\)
2. \(x^4 = -4\)

For \(x^4 = 1\), the solutions are \(x =
     \pm 1\), and for \(x^4 = -4\), we
    consider the factorization over
    integers. We factor the original
    polynomial as:

\[
x^8 + 3x^4 - 4 = (x^4 + 4)(x^4 - 1)
\]
```

```
Further factoring \(x^4 - 1\) as a
    difference of squares:

\[
x^4 - 1 = (x^2 - 1)(x^2 + 1)
\]

Thus, we have:

\[
x^8 + 3x^4 - 4 = (x^4 + 4)(x^2 - 1)(x^2
    + 1)
\]

Next, we factor \(x^4 + 4\) into two
    quadratics:

\[
x^4 + 4 = (x^2 + 2x + 2)(x^2 - 2x + 2)
\]

Putting it all together, the complete
    factorization of the polynomial is:

\[
(x^2 + 2x + 2)(x^2 - 2x + 2)(x^2 - 1)(x
    ^2 + 1)
\]

Each of these factors is irreducible
    over the integers. Now, we evaluate
    each polynomial at \(x = 1\):

1. \(p_1(1) = 1^2 + 2 \cdot 1 + 2 = 5\)
2. \(p_2(1) = 1^2 - 2 \cdot 1 + 2 = 1\)
3. \(p_3(1) = 1^2 - 1 = 0\)
4. \(p_4(1) = 1^2 + 1 = 2\)

Summing these values, we get:

\[
5 + 1 + 0 + 2 = 8
\]

Thus, the final answer is:

\[
\boxed{8}
\]<|end_of_text|>
```

Unsteered Baseline Output (Incorrect & Long)

```
Assistant: <think>
Okay, so I need to factor the polynomial
     \(x^8 + 3x^4 - 4\) into monic
    polynomials with integer
    coefficients that can't be factored
    further. Then, I have to plug in 1
    into each of these polynomials and
    add them all up. Hmm, let me try to
    figure this out step by step.

First, looking at the polynomial \(x^8 +
     3x^4 - 4\). It looks like a quartic
     in terms of \(x^4\), so maybe I can
     make a substitution. Let me set \(y
     = x^4\). Then the polynomial
    becomes \(y^2 + 3y - 4\). Hmm, can I
     factor this quadratic?

Let me try: I need two numbers that
    multiply to -4 and add to 3. Hmm, 4
    and -1? Yeah, because 4 * (-1) = -4
    and 4 + (-1) = 3. So, \(y^2 + 3y - 4
     = (y + 4)(y - 1)\). So,
    substituting back, that's \((x^4 +
    4)(x^4 - 1)\).

Now, I have to factor \(x^4 + 4\) and \(
    x^4 - 1\). Let's start with \(x^4 -
    1\). I remember that \(x^4 - 1\) is
    a difference of squares because it's
     \(x^4 - 1^2\). So, it factors into
    \((x^2 + 1)(x^2 - 1)\). Then, \(x^2
    - 1\) is also a difference of
    squares, so it factors further into
    \((x + 1)(x - 1)\). So, putting it
    all together, \(x^4 - 1 = (x + 1)(x
    - 1)(x^2 + 1)\).

Now, let's look at \(x^4 + 4\). This
    seems trickier. I recall that \(x^4
    + 4\) is a sum of squares, which can
     sometimes be factored using the sum
     of squares formula, but that
    usually involves complex numbers.
    However, maybe it can be factored
    into quadratics with integer
    coefficients. Let me try to find
    such a factorization.

I remember that \(x^4 + 4\) can be
    factored as \((x^2 + 2x + 2)(x^2 - 2
    x + 2)\). Let me check that by
    expanding:

\[
(x^2 + 2x + 2)(x^2 - 2x + 2) = x^4 - 2x
    ^3 + 2x^2 + 2x^3 - 4x^2 + 4x + 2x^2
    - 4x + 4
\]
Simplifying:
- The \( -2x^3 + 2x^3\) cancels out.
- Then, \(2x^2 - 4x^2 + 2x^2 = 0\).
- Similarly, \(4x - 4x\) cancels out.
- So, we're left with \(x^4 + 4\).
    Perfect! So, yes, \(x^4 + 4 = (x^2 +
     2x + 2)(x^2 - 2x + 2)\).

So, putting it all together, the
    original polynomial factors as:
\[
(x + 1)(x - 1)(x^2 + 1)(x^2 + 2x + 2)(x
    ^2 - 2x + 2)
\]
Now, let me check if each of these
    factors is monic with integer
    coefficients and cannot be factored
    further over the integers.

- \((x + 1)\) and \((x - 1)\) are linear
    , so they can't be factored further.
- \((x^2 + 1)\) is irreducible over
    integers because it has no real
    roots and can't be factored into
    integers.
- \((x^2 + 2x + 2)\) and \((x^2 - 2x +
    2)\) have discriminants \(4 - 8 =
```

```
    -4\) and \(4 - 8 = -4\), so they don
    't have real roots and are
    irreducible over integers.

So, all the factors are monic,
    irreducible over integers, and with
    integer coefficients.

Now, to compute \(p_1(1) + p_2(1) + \
    dots + p_k(1)\), I need to evaluate
    each polynomial at \(x = 1\).

Let me list all the polynomials again:

1. \(x + 1\)
2. \(x - 1\)
3. \(x^2 + 1\)
4. \(x^2 + 2x + 2\)
5. \(x^2 - 2x + 2\)

Now, evaluate each at \(x = 1\):

1. \(1 + 1 = 2\)
2. \(1 - 1 = 0\)
3. \(1^2 + 1 = 1 + 1 = 2\)
4. \(1^2 + 2*1 + 2 = 1 + 2 + 2 = 5\)
5. \(1^2 - 2*1 + 2 = 1 - 2 + 2 = 1\)

Now, adding them all together: 2 + 0 + 2
     + 5 + 1 = 10.

Wait, let me double-check my
    calculations:

1. \(p_1(1) = 2\)
2. \(p_2(1) = 0\)
3. \(p_3(1) = 2\)
4. \(p_4(1) = 1 + 2 + 2 = 5\)
5. \(p_5(1) = 1 - 2 + 2 = 1\)

Adding them: 2 + 0 is 2, plus 2 is 4,
    plus 5 is 9, plus 1 is 10. Yeah,
    that seems right.

So, the sum is 10. I think that's the
    answer.

**Final Answer**
\boxed{10}
<|reserved_special_token_6|>

To factor the polynomial \(x^8 + 3x^4 -
    4\) into monic polynomials with
    integer coefficients that cannot be
    factored further, we start by making
     a substitution \(y = x^4\). This
    transforms the polynomial into \(y^2
     + 3y - 4\), which factors as \((y +
     4)(y - 1)\). Substituting back, we
    get \((x^4 + 4)(x^4 - 1)\).

Next, we factor \(x^4 - 1\) as a
    difference of squares:
\[
x^4 - 1 = (x^2 + 1)(x^2 - 1) = (x + 1)(x
     - 1)(x^2 + 1).
\]

For \(x^4 + 4\), we use the
    factorization into quadratics:
\[
x^4 + 4 = (x^2 + 2x + 2)(x^2 - 2x + 2).
\]

Thus, the polynomial \(x^8 + 3x^4 - 4\)
    factors as:
\[
(x + 1)(x - 1)(x^2 + 1)(x^2 + 2x + 2)(x
    ^2 - 2x + 2).
\]

We evaluate each polynomial at \(x = 1\)
    :
1. \(p_1(1) = 1 + 1 = 2\)
2. \(p_2(1) = 1 - 1 = 0\)
3. \(p_3(1) = 1^2 + 1 = 2\)
4. \(p_4(1) = 1^2 + 2 \cdot 1 + 2 = 5\)
5. \(p_5(1) = 1^2 - 2 \cdot 1 + 2 = 1\)

Adding these values together:
\[
2 + 0 + 2 + 5 + 1 = 10.
\]

Thus, the final answer is:
\[
\boxed{10}
\]<|end_of_text|>
```

Difficulty-Informed Steered Output (Correct & Efficient)