Controllable LLM Reasoning via Sparse Autoencoder-Based Steering

Anonymous ACL submission

Abstract

Large Reasoning Models (LRMs) exhibit human-like cognitive reasoning strategies (*e.g.*, backtracking, cross-verification) during reasoning process, which improves their performance on complex tasks. Currently, reasoning strategies are autonomously selected by LRMs themselves. However, such autonomous selection often produces inefficient or even erroneous reasoning paths. To make reasoning more reliable and flexible, it is important to develop methods for controlling reasoning strategies. Existing methods struggle to control fine-grained reasoning strategies due to conceptual entanglement in LRMs' hidden states. To address this, we leverage Sparse Autoencoders (SAEs) to decompose strategy-entangled hidden states into a disentangled feature space. To identify the few strategy-specific features from the vast pool of SAE features, we propose SAE-Steering, an efficient two-stage feature identification pipeline. SAE-Steering first recalls features that amplify the logits of strategy-specific keywords, filtering out over 99% of features, and then ranks the remaining features by their control effectiveness. Using the identified strategy-specific features as control vectors, SAE-Steering outperforms existing methods by over 15% in control effectiveness. Furthermore, controlling reasoning strategies can redirect LRMs from erroneous paths to correct ones, achieving a 7% absolute accuracy improvement.

1 Introduction

Large Reasoning Models (LRMs), such as GPT-o1 (OpenAI, 2025) and DeepSeek-R1 (Guo et al., 2025), employ a "think-then-answer" paradigm, explicitly generating intermediate reasoning processes before deriving final answers. Within these reasoning processes, LRMs exhibit human-like cognitive reasoning strategies such as self-correction and cross-verification (Gandhi et al., 2025; Marjanović et al., 2025; Pan et al., 2025). Such reasoning strategies improve the accuracy and robustness of LRMs on challenging tasks (Snell et al., 2025; Zaremba et al., 2025). These LRMs autonomously select reasoning strategies during reasoning. However, such autonomous reasoning often produces inefficient or even erroneous reasoning paths (Chen et al., 2025; Wang et al., 2025). To improve the reliability and flexibility of reasoning, external guidance is promising. For example, as illustrated in Figure 1, if an LRM misinterprets the problem but pursues a flawed verification path, external guidance can redirect it to re-examine the problem statement, correcting the error. Therefore, developing methods for deliberate control over reasoning strategies is crucial.

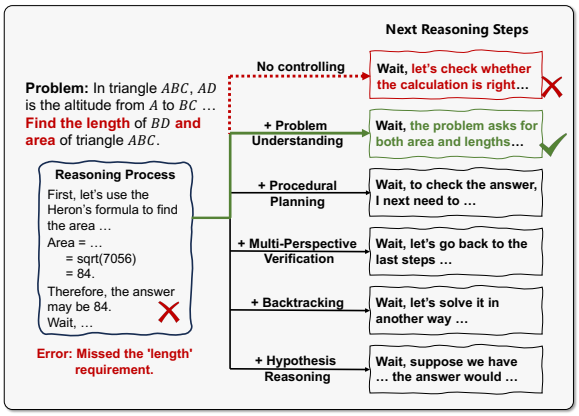

Figure 1: An illustration of reasoning strategy control. By deliberately controlling the LRM's strategy selection, we can flexibly intervene and correct its reasoning path when a flaw emerges.

Existing control methods fall into two categories: prompt-based and activation-based. Prompt-based methods control the LRM's reasoning by incorporating instructions either in the initial user prompt (Zhou et al., 2024) or during intermediate reasoning stages (Wu et al., 2025; Zhang et al., 2025). However, these methods lack direct control over the LRM's internal generative process, which results in frequent instruction-following failures, es-

pecially when reasoning context is long or instructions conflict with pre-trained behaviors (Qi et al., 2025). Activation-based methods offer more direct control by deriving a control vector to modify the LRM's hidden states during generation (Venhoff et al., 2025). This control vector is typically computed as activation differences between contrastive pairs exhibiting or lacking a target behavior (Tang et al., 2025). However, curating contrastive pairs that cleanly isolate a single strategy is difficult. As a result, the derived control vectors are prone to concept entanglement (Elhage et al., 2022; Yang et al., 2025b), inadvertently capturing features of multiple strategies and hindering precise control.

To overcome this limitation, we propose leveraging Sparse Autoencoders (SAEs) (Huben et al., 2024) to decompose the LRM's hidden states into a sparse set of interpretable and monosemantic features (Bricken et al., 2023). Specifically, a well-trained SAE projects the low-dimensional, strategy-entangled hidden states of an LRM into a high-dimensional, disentangled feature space. This projection aims to isolate strategy-specific features in the high-dimensional space, thereby providing disentangled control vectors for reasoning strategy control. However, the high-dimensional feature space introduces a new challenge: identifying the few strategy-specific features from tens of thousands of learned SAE features. Existing selection methods (Galichin et al., 2025), which rely on differential activation strength across contrastive pairs, face the same difficulty in constructing clean contrastive pairs. Furthermore, high activation does not guarantee effective control, leading to the selection of many spurious or ineffective features.

To address this, we propose identifying effective features by directly assessing their capacity to steer target strategy generation. Considering exhaustively evaluating all features is computationally infeasible, we introduce **SAE-Steering**, a two-stage pipeline for efficiently identifying and selecting effective strategy control features, balancing cost and precision. As shown in Figure 2, SAE-Steering first employs a low-cost, high-recall criterion to rapidly filter out over 99% of irrelevant features by identifying those that amplify the logits of strategy-specific keywords—a strong indicator of control potential. It then applies a more computationally intensive evaluation to quantitatively assess and rank the control effectiveness of remaining candidates on a small validation set, selecting the most effective features for final application. Extensive evaluations demonstrate that SAE-Steering consistently outperforms baselines by over 15% in control effectiveness across various reasoning tasks and LRM architectures. Moreover, SAE-Steering can correct erroneous reasoning paths in LRMs, improving absolute accuracy by 7%, highlighting the potential of strategic control. In summary, the contributions of this work are threefold:

- We leverage SAEs to disentangle and identify strategy-specific features, overcoming the concept entanglement problem inherent in controlling reasoning strategies.
- We propose SAE-Steering to identify strategy-specific features, addressing the challenge of efficient and effective feature selection from the massive set of SAE features.
- Extensive experiments validate SAE-Steering's effectiveness and robustness in controlling reasoning strategies and demonstrate its potential use in correcting erroneous reasoning paths.

2 Preliminary

Strategy Selection. LRMs employ a diverse range of cognitive reasoning strategies during their reasoning processes, making a comprehensive evaluation of control over each one impractical. Therefore, we focus on five representative reasoning strategies that are frequent, effective, and widely studied in prior work (Gandhi et al., 2025; Zhong et al., 2024). As illustrated in Figure 1, the five strategies we selected are:

- **Problem Understanding**: rephrasing the problem statement, clarifying its constraints and interpreting the given information.
- **Procedural Planning**: defining a sub-task or outlining a plan for the subsequent reasoning.
- **Backtracking**: identifying a mistake in previous reasoning and attempting to correct it or revert to a prior step.
- **Multi-Perspective Verification**: verifying a conclusion by applying a different method or examining specific cases.
- **Hypothesis Reasoning**: making an assumption or posing a "what if" scenario to explore possibilities or test certain conditions.

Importantly, this selection is purely for evaluation convenience; our method is general and applicable to control other reasoning strategies as well.

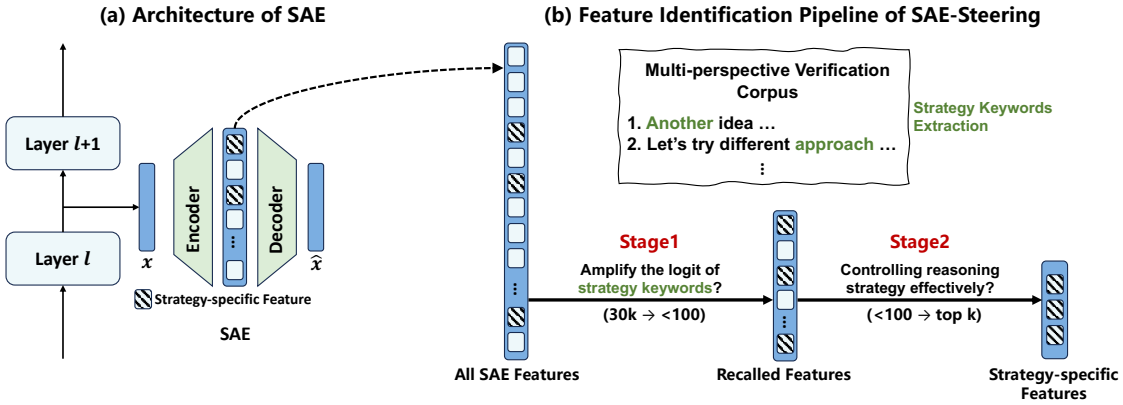

Figure 2: (a) Overview of the SAE architecture. (b) Feature identification pipeline of SAE-Steering. Numbers below the arrows indicate the approximate count of features retained.

Task Formulation. We next formalize the task of controlling reasoning strategies. In a standard autoregressive setting, an LRM generates the next token y_t based on the prefix $Y_{<t} = \{y_1, \ldots, y_{t-1}\}$. The LRM processes $Y_{<t}$ through its L transformer layers, producing a sequence of residual stream activations $\{\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^L\}$. In vanilla decoding, these activations remain unmodified. Strategy control departs from this by injecting a control vector $\Delta\mathbf{x}^\ell$ at a specific layer ℓ:

$$\mathbf{x}_t'^\ell = \mathbf{x}_t^\ell + \alpha \cdot \Delta\mathbf{x}^\ell, \tag{1}$$

where $\alpha \in \mathbb{R}$ is a coefficient controlling the steering strength. The activation $\mathbf{x}_t'^\ell$ then replaces $\mathbf{x}_t^\ell$ and is propagated through the remaining layers, influencing the final generation. By repeating this intervention for T consecutive tokens, the LRM produces a steered trajectory $Y' = \{y'_t, y'_{t+1}, \ldots, y'_{t+T-1}\}$. Given a pre-specified reasoning strategy s, the goal of reasoning strategy control is to construct $\Delta\mathbf{x}^\ell$ such that the steered trajectory Y' exhibits the desired strategy s.

3 Method

This section details our method in two parts. First, we describe how we control reasoning strategies by manipulating strategy-specific features identified in the SAE (Section 3.1). Second, we introduce SAE-Steering, a two-stage pipeline developed to effectively identify these features from the vast SAE feature pool (Section 3.2).

3.1 Strategy Control with SAE Features

We train SAEs to disentangle and identify strategy-specific features, which then serve as the control vectors for strategy control. As illustrated in Figure 2a, an SAE is an encoder–decoder architecture trained to represent an input activation as a sparse linear combination of learned feature directions. Given a residual stream activation $\mathbf{x} \in \mathbb{R}^N$, it encodes $\mathbf{x}$ into a sparse feature activation vector $\mathbf{z} \in \mathbb{R}^M$ ($M \gg N$) and reconstructs it as $\hat{\mathbf{x}}$:

$$\mathbf{z} = \sigma\left(\mathbf{W}_{\text{enc}}(\mathbf{x} - \mathbf{b}_{\text{dec}}) + \mathbf{b}_{\text{enc}}\right), \tag{2}$$

$$\hat{\mathbf{x}} = \mathbf{W}_{\text{dec}}\mathbf{z} + \mathbf{b}_{\text{dec}}, \tag{3}$$

where $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{M\times N}$, $\mathbf{b}_{\text{enc}} \in \mathbb{R}^M$, $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{N\times M}$, $\mathbf{b}_{\text{dec}} \in \mathbb{R}^N$, and σ is an activation function.

The SAE is trained to satisfy a dual objective: (1) minimizing the reconstruction error $\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$ and (2) enforcing a sparsity restriction, which dictates that the reconstruction must be constructed from only a few active latent directions[1]. This training process enables the SAE to approximate $\mathbf{x}$ as a sparse linear combination of the decoder columns:

$$\mathbf{x} \approx \mathbf{b}_{\text{dec}} + \sum_{i=1}^{M} z_i(\mathbf{x})\mathbf{f}_i \tag{4}$$

where each column $\mathbf{f}_i$ of $\mathbf{W}_{\text{dec}}$ corresponds to a disentangled and interpretable latent direction, which we refer to as a *feature* throughout the paper. The scalar $z_i(\mathbf{x})$ is the i-th component of the activation vector $\mathbf{z}$, indicating the activation strength of each feature for the input $\mathbf{x}$.

A key benefit of this decomposition is that the sparsity objective encourages *monosemanticity* (Bricken et al., 2023): each learned feature tends to capture a single concept, significantly mitigating the concept entanglement (Huben et al., 2024).

[1]We enforce sparsity via a Top-K activation function, which only retains the K largest activation values and sets the rest to zero, following (Gao et al., 2025).

We then identify the strategy-specific feature $\mathbf{f}_s$ (one of the learned $\mathbf{f}_i$ directions) that is associated with the target reasoning strategy s (see identification methods in Section 3.2). By using $\mathbf{f}_s$ as the control vector $\Delta\mathbf{x}$ in Eq. 1, we steer the LRM's reasoning strategy by repeatedly injecting $\mathbf{f}_s$ into the residual stream activations at the SAE-trained layer ℓ for the next T tokens generation:

$$\mathbf{x}'^{\ell}_{t+k} = \mathbf{x}^{\ell}_{t+k} + \alpha \cdot \mathbf{f}_s, \quad k = 0, 1, \ldots, T-1 \tag{5}$$

where α is the steering strength. The selection of α is a trade-off: excessively large values cause repetitive outputs (Fu et al., 2021), while excessively small values fail to control effectively. For each feature, we determine α by searching downwards from an empirically chosen high value, iteratively decreasing it until repetitive generation is eliminated (see Appendix A for details).

3.2 Identification of Strategy-specific Features

To efficiently identify the few critical, strategy-specific features from tens of thousands of learned SAE features, we introduce SAE-Steering, a two-stage pipeline designed for both efficiency and precision. The first stage employs a low-cost, high-recall criterion to rapidly construct a compact candidate set, while the second stage applies a more computationally intensive, high-fidelity evaluation to select the most effective features. As shown in Figure 2b, SAE-Steering first recalls features that amplify the logits of strategy-specific keywords. This stage is low-cost and highly-efficient, filtering out 99% irrelevant features. Subsequently, SAE-Steering evaluates and ranks the control effectiveness of remaining candidates on a small validation set, selecting top-ranked feature for application.

Stage 1: Recall based on logit estimation. In the first stage, we efficiently distill a small set of promising candidates from tens of thousands of SAE features by selecting those that positively influence the logits of strategy keywords. The guiding hypothesis is that features which substantially increase these keyword logits are more likely to steer the LRM toward the corresponding reasoning strategy.

Specifically, we first extract strategy keywords following the approach of Galichin et al. (2025). These keywords serve as a computationally efficient proxy to identify features potentially correlated with the target strategy. Briefly, we first create a strategy-specific corpus by manually identifying reasoning segments in the LRM's responses. We then extract the most frequent words from each corpus to serve as strategy keywords (see Appendix B for the keywords list and identification details).

Next, we estimate all SAE features' potential logit contribution to strategy keywords using logit lens (nostalgebraist, 2020). Logit lens is a method commonly used to estimate the logit contribution of hidden state activations to each token in the vocabulary. We adapt it to SAE features as follows:

Formally, let $\mathbf{U} \in \mathbb{R}^{N \times V}$ be the LRM's unembedding matrix (*i.e.,* the weight matrix of the LM head), mapping hidden activations to logits over a vocabulary of size V. Let $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{N \times M}$ be the SAE decoder matrix. As described in Section 3.1, each column of $\mathbf{W}_{\text{dec}}$ corresponds to a disentangled feature direction $\mathbf{f}_i \in \mathbb{R}^N$. We compute the logit contribution matrix $\mathbf{L} \in \mathbb{R}^{M \times V}$ for all features via:

$$\mathbf{L} = \mathbf{W}_{\text{dec}}^{\top}\mathbf{U}, \tag{6}$$

where the i-th row $\mathbf{L}_{i,:}$ gives the logit contributions of feature $\mathbf{f}_i$ across the vocabulary. This computation requires only a single matrix multiplication, making it low-cost and efficient.

We aim to recall features that specifically and significantly amplify strategy keywords, while avoiding those that amplify irrelevant tokens more strongly than the keywords. To achieve this, we extract the top-10 tokens with the highest logit contribution for each feature and recall features satisfying: (i) at least n of these tokens are strategy keywords, and (ii) each such keyword's logit contribution exceeds a threshold τ. This recall step is highly selective, narrowing the candidate pool from tens of thousands of features to several tens.

Stage 2: Rank based on Control Effectiveness. In the second stage, we evaluate and rank the candidate features from Stage 1 to identify those with the highest control effectiveness. This ranking is based on their empirical performance on a small validation set $\mathcal{P}$.

Formally, for each problem $p \in \mathcal{P}$ with a given response prefix $Y_{<t}$, we generate two distinct T-token continuations[2]: (i) a baseline trajectory Y_0, generated via standard decoding, and (ii) a steered trajectory $Y^{(j)}$, generated using the candidate feature $\mathbf{f}_j$ as the control vector. An LLM judge then

[2]We set the sampling temperature to 0 to eliminate randomness as a confounding factor in our evaluation.

assesses whether $Y^{(j)}$ more explicitly demonstrates the target strategy s than Y_0 [3], yielding binary judgment $J_{p,j} \in \{0, 1\}$. The control effectiveness of a feature $\mathbf{f}_j$ is then calculated as the control success rate over the validation set:

$$\text{Effectiveness}(\mathbf{f}_j) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} J_{p,j}. \quad (7)$$

This empirical ranking allows us to select the top-ranked feature as $\mathbf{f}_s$ for the target strategy s.

4 Experiments

In this section, we conduct experiments to address the following research question:

- **RQ1:** Can our SAE-based steering method, leveraging the identified features, reliably control LRMs' reasoning strategies?
- **RQ2:** How effective is SAE-Steering for strategy-specific feature identification?
- **RQ3:** Can we correct an LRM's erroneous reasoning path by deliberately controlling its reasoning strategies?

4.1 Experiment Setup

Datasets. We train our SAEs on activations from a mixed corpus combining LMSYS-CHAT-1M (Zheng et al., 2024) and OPENTHOUGHTS-114K (Team, 2025a), following prior work (Galichin et al., 2025). For the evaluation of reasoning strategy control, we first randomly sample 50 responses from past AIME competitions (1983–2023) (AIME, 2025) as the validation set. We then evaluate control effectiveness on 200 randomly sampled responses from AIME'24 and 25 (AIME, 2025) and 200 responses from GPQA (Rein et al., 2023). GPQA is a science reasoning dataset spanning biology, physics, and chemistry, which we use to assess the out-of-domain generalization capability of our strategy-specific features.

Baselines. We compare SAE-Steering with three representative control methods:

- **Logit Boosting**, which directly boosts the logits of strategy-specific keywords;
- **Think Intervention** (Wu et al., 2025), which inserts human-crafted instructions into the middle of the reasoning process;
- **Vector Steering** (Venhoff et al., 2025), which uses an LLM to annotate reasoning strategies for constructing contrastive datasets, then extracts control vectors via contrast pairs.

Evaluation Protocol. We evaluate control effectiveness following the procedure described in Stage 2 of Section 3.2. Importantly, for feature selection in Stage 2 of SAE-Steering, we use only GPT-4o (OpenAI, 2024) as the judge. For test evaluation, we employ three LLM judges—GPT-4o (OpenAI, 2024), Gemini-2.5-flash (Comanici et al., 2025), and Deepseek-V3.2 (Liu et al., 2024)—to vote as judges. This majority voting mitigates individual judge biases and ensures more reliable evaluation. We also test the agreement between LLM judges and human annotators, which achieves a high agreement rate of 0.82 (see Appendix C for details), confirming the reliability of LLM judges.

Implementation Details. We train TopK-SAEs (Gao et al., 2025) (with $K = 50$) on the last layer of DeepSeek-R1-Distill-Llama-8B (Guo et al., 2025) (hereafter referred to as R1-Llama-8B) and Qwen3-8B (Team, 2025b). For SAE-Steering hyperparameters, we set $n = 2$ and $\tau = 0.1$ in Stage 1, and continuation length $T = 512$ in Stage 2. For sampling, we set the temperature to 0 during control effectiveness evaluations to eliminate confounding effects from sampling stochasticity. For error correction experiments, we adopt the officially recommended temperature of 0.6 and set the maximum token length to 32,768.

4.2 Control Effectiveness of SAE-Based Steering (RQ1)

SAE-based steering outperforms baselines. We report the control effectiveness of different methods in Table 1, from which we make the following observations:

(1) Activation-based methods (Vector Steering and SAE-Steering) consistently outperform prompt-based methods (Think Intervention) except in some cases within Hypothesis Reasoning, which demonstrates the superiority of directly intervening in hidden states.

(2) SAE-Steering significantly outperforms Vector Steering, with an average improvement of 15%. We attribute this to the disentangling properties of SAEs, which mitigate the conceptual entanglement present in control vectors, thereby enabling more precise strategy control.

[3] We provide the prompt and validate the reliability of LLM Judges in Appendix C.

Dataset	Method	R1-Llama-8B					Qwen3-8B					Average
		PU	PP	BK	MV	HR	PU	PP	BK	MV	HR	
AIME	Logit Boosting	0.21	0.49	0.30	0.27	0.32	0.44	0.61	0.39	0.49	0.56	0.41
	Think Intervention	0.56	0.49	0.21	0.21	0.39	0.62	0.81	0.12	0.23	0.61	0.43
	Vector Steering	0.69	0.82	0.67	0.48	0.34	0.74	0.85	0.55	0.51	0.52	0.62
	SAE-Steering	**0.88**	**0.86**	**0.69**	**0.76**	**0.41**	**0.92**	**0.92**	**0.78**	**0.70**	**0.65**	**0.76**
GPQA	Logit Boosting	0.28	0.68	0.29	0.39	0.56	0.43	0.79	0.40	0.47	0.63	0.49
	Think Intervention	0.66	0.69	0.35	0.23	0.57	0.68	0.83	0.17	0.16	0.77	0.51
	Vector Steering	0.77	0.90	0.61	0.52	0.51	0.89	0.89	0.80	0.55	0.72	0.72
	SAE-Steering	**0.94**	**0.90**	**0.78**	**0.93**	**0.70**	**0.94**	**0.95**	**0.81**	**0.82**	**0.89**	**0.87**

Table 1: Control effectiveness evaluation across five reasoning strategies: Problem Understanding (PU), Procedural Planning (PP), Backtracking (BK), Multi-Perspective Verification (MV), and Hypothesis Reasoning (HR).

(3) SAE features identified in the math domain demonstrate comparable effectiveness on scientific reasoning tasks (GPQA), indicating the generalizability of SAE features across different reasoning domains.

(4) All methods achieve better control effectiveness on GPQA than on AIME. This may because the reasoning length of GPQA is much shorter than AIME (7k vs. 15k tokens), which makes control easier.

SAE features function beyond keyword amplification. Importantly, although our SAE features are recalled by identifying features that amplify the logits of strategy-specific keywords, they encode deeper strategy concepts beyond keyword promotion. This is evidenced by SAE-Steering achieving over 35% better control effectiveness than Logit Boosting (Table 1). To illustrate this more explicitly, we present a case study in Figure 3. In this case, boosting the logits of strategy keywords like "another" does not genuinely change the reasoning strategy. The LRM generates "another" but still continues to verify its answer by testing different values of m. In contrast, SAE-Steering successfully guides the LRM to adopt the Multi-Perspective Verification strategy. Additional steering examples are available in Figure 6.

No steering:
Wait, m=132 is the first one with four primes. But let me check m=132 again. Is there a smaller m?

Wait, **m=105: 10931=17×643. Two primes**.
...

Logit Boosting:
Wait, m=132 is the first one with four primes. But **let me check m=another value**. For example, m=another number where the expression could be composite with more factors.
...

SAE-Steering:
Alternatively, maybe I can think of another approach. Maybe the original problem is similar to the Euler's prime-generating polynomial but with a different constant term
...

Figure 3: Case study: SAE-Steering changes reasoning behavior while Logit Boosting only boosts keywords.

4.3 Effectiveness of SAE-Steering for Feature Identification (RQ2)

Logit-based recall is more precise than activation-based. We compare the effectiveness of identifying strategy-specific features of SAE-Steering with existing activation-based methods, specifically ReasonScore (Galichin et al., 2025). ReasonScore recalls features based on their activation strength on keywords compared to other tokens in the reasoning context. We use ReasonScore to recall the same number of features as our Stage 1 (143 for R1-Llama-8B and 357 for Qwen3-8B) and evaluate the precision of recalled features, *i.e.,* the proportion of recalled features that successfully control reasoning strategies. As shown in Table 2, SAE-Steering outperforms ReasonScore by 28% in precision, demonstrating the superiority of logit-based over activation-based feature identification. Logits directly measure causal effects on outputs, better reflecting features' actual control capability than activation strength.

	R1-Llama-8B	Qwen3-8B
ReasonScore	0.33	0.27
SAE-Steering	**0.61**	**0.52**

Table 2: Precision of recalled features.

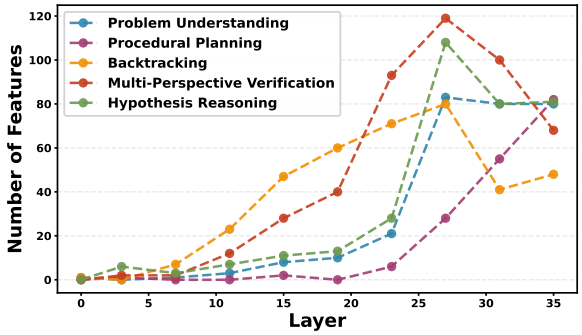

Figure 4: Recalled features across layers.

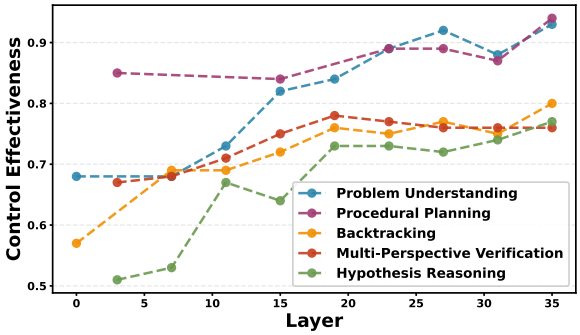

Figure 5: Control effectiveness across layers.

Layer-wise analysis of feature identification. In the main experiments, we train SAEs on the last layer of LRMs. Here we further investigate how the identification of strategy-specific features varies across layers. Due to computational constraints, we limit this analysis to Qwen3-8B. We first examine the presence of strategy-specific features across layers by measuring the number of features recalled by Stage 1 of SAE-Steering. As shown in Figure 4, strategy-specific features are rare in shallow layers (0, 3, 7, 11) but prevalent in deeper layers (23, 27, 31, 35), which is consistent with prior findings that abstract reasoning concepts are primarily encoded in the deeper layers of LRMs (Yun et al., 2021; Shi et al., 2025).

We next investigate the control effectiveness of these features across layers by reporting the average control effectiveness of the top-3 features. As shown in Figure 5, shallow layers exhibit poor control effectiveness, while layers beyond 20 demonstrate strong and relatively stable control effectiveness. This suggests that reasoning strategy control should be applied to middle-to-late layers for optimal results.

4.4 Correcting Erroneous Reasoning Paths via Strategy Control (RQ3)

Setup. To demonstrate the practical value of strategy control, we test whether controlling reasoning strategies can correct errors **even after the LRM has already generated a wrong answer**—a more challenging setting than simple generation. Specifically, we sample incorrect LRM responses on the MATH500 (Lightman et al., 2023), AIME25 (AIME, 2025), and GPQA (Rein et al., 2023), and attempt to correct them during an extended reasoning process (See Appendix D for sample details and dataset statistics). Following Budget Forcing (Muennighoff et al., 2025), we insert a "wait" token at the end of the initial, flawed reasoning to induce further thinking. During this extended reasoning phase, we apply SAE-Steering to control the LRM's subsequent reasoning strategy. To select the most appropriate strategy for different problems, we train a strategy router (Appendix E). We compare our approach with two common self-correction baselines: (1) Budget Forcing (Muennighoff et al., 2025), which only extends reasoning without strategic guidance; and (2) Self-Reflection (Shinn et al., 2023), which prompts the LRM to reflect on its previous answer and generate a new response.

Results. The error correction results are shown in Table 3, from which we make the following observations:

(1) The highest correction rate is only 33%, with MATH500 achieving the highest rate and AIME the lowest. This demonstrates the difficulty of error correction, and harder tasks are also more difficult to correct.

(2) Budget Forcing outperforms Self-Reflection on all datasets except GPQA on Qwen3-8B, demonstrating the advantage of continuous reasoning. By continuing from the current state rather than reprocessing the entire reasoning process, Budget Forcing maintains better focus on error correction.

Model	Method	MATH500	AIME25	GPQA
R1-Llama-8B	Self-Reflection	0.1394	0.0123	0.0262
	Budget Forcing	0.2121	0.0123	0.0626
	SAE-Steering	**0.3313**	**0.0552**	**0.1196**
Qwen3-8B	Self-Reflection	0.0993	0.0411	0.0749
	Budget Forcing	0.1773	0.0685	0.0484
	SAE-Steering	**0.2411**	**0.1370**	**0.1154**

Table 3: Error correction rates across methods and datasets.

(3) SAE-Steering consistently outperforms Budget Forcing across all LRMs and datasets, with an average absolute accuracy improvement of 7%. This suggests that deliberately controlling reasoning strategies enables more effective error correction.

5 Related Work

Reasoning Strategies in LRMs. Early studies attempt to improve LLM performance on complex tasks by designing prompts to guide reasoning processes (Shinn et al., 2023; Zhou et al., 2024). Recent research demonstrates that LLMs trained with rule-based reinforcement learning can unsupervisedly develop human-like cognitive reasoning strategies such as self-reflection and backtracking (Liu et al., 2024). These advancements have led to the emergence of current LRMs. During inference, LRMs produce long Chains-of-Thoughts (CoTs) that explore diverse reasoning paths while continuously verifying previous steps (Marjanović et al., 2025). In this process, LRMs employ diverse human-like cognitive reasoning strategies such as backtracking and multi-perspective verification. The use of these reasoning strategies improves their accuracy and robustness in solving complex problems (Gandhi et al., 2025; Snell et al., 2025; Muennighoff et al., 2025).

Controllable LLM Reasoning. Many works attempt to control LRM reasoning behavior. These methods can be categorized into prompt-based and activation-based. Prompt-based methods (Wu et al., 2025; Yang et al., 2025a; Zhang et al., 2025) insert human-scripted instructions into intermediate reasoning steps, mimicking the LRM's style to seamlessly steer its reasoning trajectory. Activation-based methods directly modify hidden states using control vectors derived from contrastive activation analysis. For example, many works (Sheng et al., 2025; Tang et al., 2025; Lin et al., 2025) obtain control vectors by contrasting activations between short and long CoT responses. However, such pairs fail to isolate individual strategies, causing control vectors to suffer from concept entanglement and only enable coarse-grained control (*e.g.,* reasoning length) rather than fine-grained strategy control. Venhoff et al. (2025) address this by using LLM judges to annotate each reasoning step with fine-grained strategy labels, then contrasting activations across labels. However, accurate step-level annotation is challenging. Conversely, we leverage SAEs to learn monosemantic features in an unsupervised way, eliminating annotation requirements while better disentangling conceptually-entangled hidden states.

Sparse Autoencoders. Mechanistic interpretability seeks to understand the internal workings of LRMs by analyzing the structure and function of their learned representations (Singh et al., 2024; Gantla, 2025). A primary tool in this field is SAEs, which decompose high-dimensional LRM activations into a sparse set of latent features (Bricken et al., 2023; Huben et al., 2024). These features often correspond to human-interpretable concepts, enabling researchers to probe and manipulate specific aspects of LRM behavior (Deng et al., 2025; Yang et al., 2025b). For example, (Galichin et al., 2025) leveraged SAEs to identify features associated with reasoning. In their method, reasoning features are selected as those that activate more strongly on reasoning-related keywords (*e.g.,* 'wait', 'alternatively') than on other tokens. However, high activation strength does not necessarily indicate control capacity, causing such methods to recall many features that show superficial correlations with reasoning behaviors but lack the ability to effectively control fine-grained reasoning strategies. Instead, we recall features through their direct logit contributions to strategy-specific tokens, enabling more precise recall of features with genuine control effectiveness.

6 Conclusion

In this work, we leverage strategy-specific features of SAEs to achieve fine-grained control over LRMs' reasoning strategies. SAEs decompose strategy-entangled hidden states into disentangled strategy-specific features. To identify these strategy-specific features from the vast pool of SAE features, we propose SAE-Steering, a two-stage feature identification pipeline that balances efficiency and precision. SAE-Steering first employs a logit estimation method to rapidly recall candidate features that amplify strategy-specific keywords, then ranks the control effectiveness of remaining features through intervention experiments on a validation set. Extensive experiments demonstrate the effectiveness and robustness of our identified features in controlling reasoning strategies. Furthermore, we demonstrate that controlling reasoning strategies can redirect LRMs from erroneous paths to correct ones.

Limitations

While SAE-Steering demonstrates promising results, several limitations remain to be addressed in future work. First, due to computational constraints, we only evaluated five representative strategies to demonstrate our method's effectiveness. Future work could investigate controlling other reasoning strategies. Second, we only demonstrate the application of controlling reasoning strategies in error correction scenarios. Future work could explore applying such control to a wider range of applications. Third, we only attempted to correct erroneous reasoning paths by enforcing LRMs to continue reasoning and controlling subsequent reasoning strategies. Future work could explore guiding the LRM at earlier stages—either at the beginning or during intermediate steps—to dynamically adjust the reasoning trajectory.

References

AIME. 2025. Aime problems and solutions.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *ICML*. OpenReview.net.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Boyi Deng, Yu Wan, Baosong Yang, Yidan Zhang, and Fuli Feng. 2025. Unveiling language-specific features in large language models via sparse autoencoders. In *ACL (1)*, pages 4563–4608. Association for Computational Linguistics.

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

Zihao Fu, Wai Lam, Anthony Man-Cho So, and Bei Shi. 2021. A theoretical analysis of the repetition problem in text generation. In *AAAI*.

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

Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *Second Conference on Language Modeling*.

Sandeep Reddy Gantla. 2025. Exploring mechanistic interpretability in large language models: Challenges, approaches, and insights. In *2025 International Conference on Data Science, Agents & Artificial Intelligence (ICDSAAI)*.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. Scaling and evaluating sparse autoencoders. In *ICLR*.

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

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *ICLR*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*. Association for Computational Linguistics.

Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. 2024. Numinamath.

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*.

Zhengkai Lin, Zhihang Fu, Ze Chen, Chao Chen, Liang Xie, Wenxiao Wang, Deng Cai, Zheng Wang, and Jieping Ye. 2025. Controlling thinking speed in reasoning models. *NeurIPS*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Sara Vera Marjanović, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, Nicholas Meade, Dongchan Shin, Amirhossein Kazemnejad, Gaurav Kamath, Marius Mosbach, Karolina Stańczak, and Siva Reddy. 2025. Deepseek-r1 thoughtology: Let's think about llm reasoning. *arXiv preprint arXiv:2504.07128*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel J. Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

nostalgebraist. 2020. Interpreting gpt: The logit lens. *Less-Wrong (blog)*.

OpenAI. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI. 2025. Openai o1 system card. Accessed: 2025-02-21.

Qianjun Pan, Wenkai Ji, Yuyang Ding, Junsong Li, Shilian Chen, Junyi Wang, Jie Zhou, Qin Chen, Min Zhang, Yulan Wu, and Liang He. 2025. A survey of slow thinking-based reasoning llms using reinforced learning and inference-time scaling law. *arXiv preprint arXiv:2505.02665*.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Amy Xin, Youfeng Liu, Bin Xu, Lei Hou, and Juanzi Li. 2025. AGENTIF: benchmarking instruction following of large language models in agentic scenarios. *CoRR*, abs/2505.16944.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Leheng Sheng, An Zhang, Zijian Wu, Weixiang Zhao, Changshuo Shen, Yi Zhang, Xiang Wang, and Tat-Seng Chua. 2025. On reasoning strength planning in large reasoning models. *NeurIPS*.

Wei Shi, Sihang Li, Tao Liang, Mingyang Wan, Gojun Ma, Xiang Wang, and Xiangnan He. 2025. Route sparse autoencoder to interpret large language models. *CoRR*, abs/2503.08200.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *NeurIPS*.

Chandan Singh, Jeevana Priya Inala, Michel Galley, Rich Caruana, and Jianfeng Gao. 2024. Rethinking interpretability in the era of large language models. *arXiv preprint arXiv:2402.01761*.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *ICLR*. OpenReview.net.

Xinyu Tang, Xiaolei Wang, Zhihao Lv, Yingqian Min, Xin Zhao, Binbin Hu, Ziqi Liu, and Zhiqiang Zhang. 2025. Unlocking general long chain-of-thought reasoning capabilities of large language models via representation engineering. In *ACL (1)*, pages 6832–6849. Association for Computational Linguistics.

Openthoughts Team. 2025a. Openthoughts: Data recipes for reasoning models.

Qwen Team. 2025b. Qwen3 technical report.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. 2025. Understanding reasoning in thinking language models via steering vectors. *CoRR*, abs/2506.18167.

Yue Wang, Qiuzhi Liu, Jiahao Xu, Tian Liang, Xingyu Chen, Zhiwei He, Linfeng Song, Dian Yu, Juntao Li, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Thoughts are all over the place: On the underthinking of o1-like llms. *arXiv preprint arXiv:2501.18585*.

Tong Wu, Chong Xiang, Jiachen T. Wang, and Prateek Mittal. 2025. Effectively controlling reasoning models through thinking intervention. *arXiv preprint arXiv:2503.24370*.

Chenxu Yang, Qingyi Si, Mz Dai, Dingyu Yao, Mingyu Zheng, Minghui Chen, Zheng Lin, and Weiping Wang. 2025a. Test-time prompt intervention.

Jingyuan Yang, Rongjun Li, Weixuan Wang, Ziyu Zhou, Zhiyong Feng, and Wei Peng. 2025b. Lf-steering: Latent feature activation steering for enhancing semantic consistency in large language models. *arXiv preprint arXiv:2501.11036*.

Zeyu Yun, Yubei Chen, Bruno A. Olshausen, and Yann LeCun. 2021. Transformer visualization via dictionary learning: contextualized embedding as a linear superposition of transformer factors. In *DeeLIO@NAACL-HLT*, pages 1–10. Association for Computational Linguistics.

Wojciech Zaremba, Evgenia Nitishinskaya, Boaz Barak, Stephanie Lin, Sam Toyer, Yaodong Yu, Rachel Dias, Eric Wallace, Kai Xiao, Johannes Heidecke, and 1 others. 2025. Trading inference-time compute for adversarial robustness. *arXiv preprint arXiv:2501.18841*.

Xingsheng Zhang, Luxi Xing, Chen Zhang, Yanbing Liu, Yifan Deng, Yunpeng Li, Yue Hu, and Chenxu Niu. 2025. Can we steer reasoning direction by thinking intervention? In *Findings of EMNLP 2025*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2024. Lmsys-chat-1m: A large-scale real-world LLM conversation dataset. In *ICLR*. OpenReview.net.

Qihuang Zhong, Kang Wang, Ziyang Xu, Juhua Liu, Liang Ding, and Bo Du. 2024. Achieving> 97% on gsm8k: Deeply understanding the problems makes llms better solvers for math word problems. *Frontiers of Computer Science*.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. SELF-DISCOVER: large language models self-compose reasoning structures. In *NeurIPS*.

A Selection of Steering Strength

The hyper-parameter α determines the steering strength during strategy control. An overly large α can cause the LRM to generate repetitive outputs, while an α that is too small may yield negligible controlling effects. We thus select an α value that is as large as possible without inducing repetitive outputs. Specifically, we use the validation set to determine α for each feature. For each validation sample, we first steer the feature with $\alpha = 15$ and check for repetitive outputs. If repetition occurs, we decrease α by one and re-steer. We repeat this process until no repetition is detected. We then use the average α across validation samples as the steering strength for the test set. The starting value of 15 was chosen empirically, as we found that higher values frequently lead to repetitive outputs for most features.

B Extraction of Strategy Keywords

To extract strategy keywords for each reasoning strategy, we first construct a corpus for each reasoning strategy by sampling the responses of the LRM to a diverse set of problems and manually identifying the segments corresponding to each reasoning strategy. From each strategy-specific corpus, we then extract the top-20 most frequent words and then perform a manual curation to select the keywords we identified as most representative of the target reasoning strategy. The final keywords lists are shown in Table 4.

Reasoning Strategy	High-Frequency Keywords
Problem Understanding	problem, question, statement, reads, says
Procedural Planning	let, need, planning, decomposition
Backtracking	earlier, previous, initial, back
Multi-Perspective Verification	another, example, case, approach
Hypothesis Reasoning	maybe, perhaps, assume, suppose, if

Table 4: High-frequency keywords corresponding to each reasoning strategy.

C Reliability of LLM Judges

To validate LLM judge reliability, we conducted a human annotation study. Specifically, we randomly sampled 200 steered outputs (40 per strategy) alongside their unsteered baselines. We then asked three human annotators (Krippendorff's alpha = 0.78) to evaluate whether the steered output more explicitly demonstrates the target strategy than the baseline. Taking human judgments as ground truth, we evaluate the accuracy of LLM judges. As shown in Table 5, LLM judges achieve 0.82 agreement with human annotations, indicating reliable performance.

Reasoning Strategy	Agreement
Problem Understanding	0.85
Procedural Planning	0.75
Backtracking	0.83
Multi-Perspective Verification	0.85
Hypothesis Reasoning	0.8
Average	**0.82**

Table 5: Agreement between human annotators and LLM judges.

D Curation of Error Correction Dataset

To sample incorrect LRM responses from MATH500, AIME25, and GPQA, we sample eight responses for each problem in these datasets and retain only the incorrect ones. The final dataset statistics are shown in Table 6.

Model	MATH500	AIME25	GPQA
R1-Llama-8B	495	163	878
Qwen3-8B	141	73	641

Table 6: Statistics of Error Correction Dataset.

E Strategy Router

E.1 Methods

To steer LRMs' reasoning strategies from erroneous paths to correct ones, we need to select appropriate strategies based on the current reasoning context. Reasoning strategies can be controlled either manually or by an automatic strategy router. Here we train a lightweight router via contrastive learning (van den Oord et al., 2018) to automatically select effective strategies based on the current reasoning context, thereby eliminating the need for manual intervention.

Specifically, we instantiate the strategy router as a bi-encoder architecture (Karpukhin et al., 2020). A context encoder, $E_c(\cdot)$, embeds the current reasoning state (represented by the final token of the response prefix $Y_{<t}$), and a feature encoder, $E_f(\cdot)$, projects each strategy-specific feature $\mathbf{f}_s$ into the same representation space. The effective scores between the context and a feature are then computed as the dot product of their respective embeddings:

$$\text{score}(Y_{<t}, \mathbf{f}_s) = \langle E_c(Y_{<t}),\ E_f(\mathbf{f}_s) \rangle \tag{8}$$

The router is trained using the InfoNCE loss (van den Oord et al., 2018), which encourages higher effective scores for positive context–feature pairs and lower effective scores for negative ones:

$$L(Y_{<t}, \mathbf{f}_s^+, \mathbf{f}_{s,1}^-, \ldots, \mathbf{f}_{s,M}^-) = -\log \frac{e^{\text{score}(Y_{<t}, \mathbf{f}_s^+)}}{e^{\text{score}(Y_{<t}, \mathbf{f}_s^+)} + \sum_{k=1}^{M} e^{\text{score}(Y_{<t}, \mathbf{f}_{s,k}^-)}}, \tag{9}$$

where $(Y_{<t}, \mathbf{f}_s^+)$ is labeled as a positive pair if steering with feature $\mathbf{f}_s^+$ leads to a correct final answer. All other pairings for that context are treated as negative pairs. At inference time, for a given context $Y_{<t}$, we compute $\text{score}(Y_{<t}, \mathbf{f}_s)$ for all candidate features $\mathbf{f}_s$ and select the feature with the highest effective score as the selected feature to steer the LRM.

E.2 Implementation Details

For each reasoning strategy, we select the top three strategy-specific features with the best control effectiveness on the validation set, yielding a total of fifteen features for the strategy router to choose from. We include three rather than one feature per strategy because different features may be effective in different contexts, providing the router with more flexibility to adapt to different reasoning scenarios.

For the training of our strategy router, we use a training set composed of 919 problems from past AIME competitions (1983–2023) (AIME, 2025) and 4,000 problems from the 'aops_forum' source of the NUMINAMATH-1.5 dataset (LI et al., 2024). For each problem, we sample eight initial responses. To empirically evaluate the effectiveness of a feature f_j on an incorrect response y_i, we apply reasoning steering with f_j to generate eight responses and measure the proportion of them that successfully correct the initial error. This training data is strictly separated from our test sets in Section 4.4, ensuring no data leakage. Notably, GPQA represents an out-of-domain scenario, demonstrating our method's generalization capability.

Problem:

In triangle ABC, $AB = 10$, $BC = 14$, and $CA = 16$. Let D be a point in the interior of $\overline{BC}$ .Let points I_B and I_C denote the incenters of triangles ABD and ACD, respectively. The circumcircles of triangles BI_BD and CI_CD meet at distinct points P and D. The maximum possible area of $\triangle BPC$ can be expressed in the form $a - b\sqrt{c}$,where a, b, and c are positive integers and c is not divisible by the square of any prime. Find $a + b + c$.

No steering extended:

Let me recall that the area of triangle ABC is 40√3. If the maximum area of BPC is 40√3 - 14√3 = 26√3, then a = 40, b = 14, c = 3, giving 40 + 14 + 3 = 57. But why would the area of BPC be 26√3?
...

+ Problem Understanding:

Let me think again. **The problem says** "the maximum possible area of triangle BPC can be expressed in the form a - b√c". Let me consider that the area might be related to the area of ABC minus some value.
...

+ Procedural Planning:

Let me check one more thing. The area of triangle BPC when P is at (7, 7√3) is 49√3, but the area of triangle ABC is 40√3. Since P is outside, it's possible ...
Let me try to calculate the area of BPC when P is the incenter of ABC, which is at (4, 2√3). The area is 14√3, as calculated earlier. This is 14√3, which is 14√3 - 0, but again b must be positive ...
Let me recall that in triangle BPC, with ∠BPC = π/3, and BC = 14 ...

+ Backtracking:

Actually, **let me recall that in the initial calculation**, the area of triangle ABC is 40√3. If the maximum area of BPC is 40√3, it would be equal to the area of ABC, but this is not possible since P is a point outside or inside.
...

+ Multi-perspective Verification:

Let me think about another approach.

Let me use the fact that the area of BPC is (1/2)*BP*CP*sin(∠BPC), and from the earlier relation, if ∠BPC is 2π/3, then the maximum BP*CP is 392/5, so the area is (392/5)*(√3/2) = 196√3/5. But 196/5 is 39.2, so 39.2√3 is not in the form a - b√c.

Alternatively, if the maximum area is when P is the incenter of ABC, the area is 14√3, but this is also not in the required form.

Another idea: maybe the maximum area is the area of ABC, which is 40√3, but the problem says "maximum possible", so it might be larger.
...

+ Hypothesis Reasoning:

Let me think again. **If** the area is 49√3, and **if** the problem says it can be expressed as a - b√c, **perhaps** there is a different expression. Let me recall that 49√3 is equal to 49√3 + 0, but if there is a different expression, like if the area is 49√3 - something, but I need to think.

Figure 6: A case study of SAE-based steering. By steering with different strategy-specific features as control vectors, we steer the subsequent reasoning trajectory to follow different reasoning strategies.

You are a top-tier AI recognition reasoning behavior analysis expert. Your task is to precisely evaluate the success of a thought intervention experiment. An intervention is successful if the intervened thinking **more explicitly and significantly demonstrates** the "target cognitive reasoning strategy". If the baseline thinking lacks this strategy, it must be **added**; if the baseline already exhibits it, it must be **strengthened**.

You can first count the number of times the "target cognitive reasoning strategy" appears in both the baseline thinking and the intervened thinking, and then determine whether the frequency increases after intervention. Your output **MUST** be a single valid JSON object. Provide the following:
- "before": integer, the count of occurrences in the Before Intervention text.
- "after": integer, the count of occurrences in the After Intervention text.
- "more_frequent": boolean, true if the count **after** > **before**, else false.

Target Cognitive Reasoning Strategy
{reasoning_strategy_description}

Examples:

{few_shot}

FINAL TASK

Reasoning Texts to Analyze:

Before Intervention:
{before_text}

After Intervention:
{after_text}

Your Answer:

Figure 7: The prompt used to evaluate the control effectiveness.