

ReflCtrl: Controlling LLM Reflection Efficiently via Representation Engineering

Ge Yan

CSE, UCSD

geyan@ucsd.edu

Chung-En Sun

CSE, UCSD

cesun@ucsd.edu

Linbo Liu

UCSD

linbo@ucsd.edu

Tsui-Wei (Lily) Weng

HDSI, UCSD

lweng@ucsd.edu

Abstract

Large reasoning models achieve strong performance on diverse tasks by producing extended chains of thought. **Self-reflection**, the ability to review and revise prior reasoning steps, is widely regarded as a key contributor to this performance. However, self-reflection also incurs substantial inference cost, and its governing mechanism remains underexplored. In this work, we study self-reflection through the lens of representation engineering. First, we identify a *reflection direction* in the model’s latent space that separates reflection steps from non-reflection steps, and show that activation along this direction is strongly predictive of answer correctness, suggesting that self-reflection is regulated by the model’s internal uncertainty. Next, building on this insight, we propose **ReflCtrl**, a framework that controls self-reflection via a *stepwise steering method*: interventions are applied only at the start of each new reasoning step, enabling fine-grained control over reflection frequency without degrading generation quality. Experiments across math and general reasoning benchmarks show that reflection is often redundant, especially in stronger models: ReflCtrl reduces total reasoning tokens by up to 43.2% while preserving accuracy, and substantially outperforms the conventional approach that steers at every token, at matched token budgets.

1 Introduction

Reasoning language models, such as OpenAI o1 (OpenAI, 2024) and DeepSeek-R1 (DeepSeekAI et al., 2025), achieve strong performance on diverse tasks including mathematics, coding, and general reasoning by producing extended chains of thought before a final answer. A distinctive behavior of these models, emerging from training rather than explicit prompting, is **self-reflection**: the ability to spontaneously review and revise prior reasoning steps during inference. While self-reflection is widely regarded as a key contributor to reasoning quality, it also represents a substantial inference overhead: our analysis shows reflection steps account for 23–36% of total reasoning tokens (see Fig. 2a), varying with benchmark difficulty.

Despite this prevalence and cost, the mechanism governing self-reflection is less explored. Two fundamental questions remain open.

1. **(RQ1)** *When does the model decide to initiate reflection?* Without identifying the internal trigger, it is unclear whether reflection is a principled response to uncertainty or a spurious training artifact.
2. **(RQ2)** *Is reflection always necessary?* If many reflection steps are redundant, suppressing them could substantially reduce inference cost, but only with a reliable mechanism to identify and control reflection.

Code: <https://github.com/Trustworthy-ML-Lab/ReflCtrl>

Project website: <https://lilywenglab.github.io/ReflCtrl/>

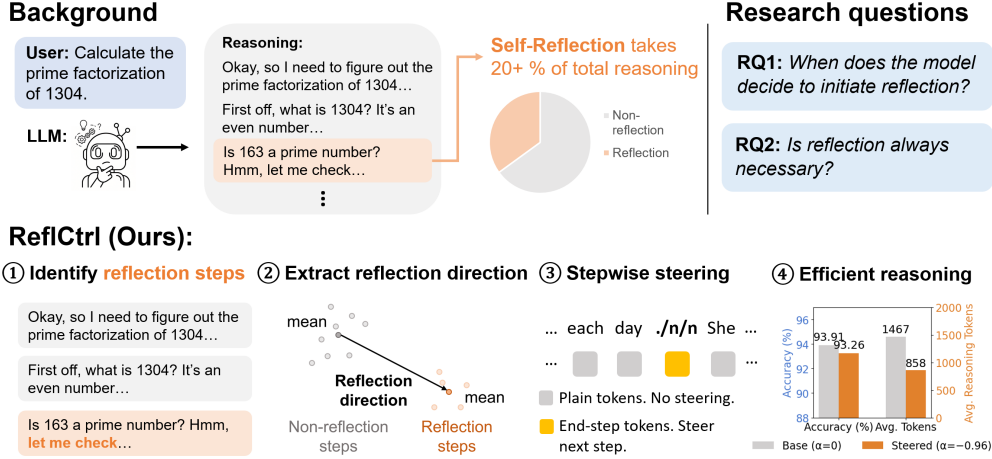


Figure 1: **Overview of the proposed ReflCtrl framework.** (1) Reflection steps are identified from the model’s chain-of-thought reasoning. (2) A reflection direction is extracted by computing the mean difference between reflection and non-reflection step representations in the latent space. (3) During inference, steering is applied only at end-of-step tokens (e.g., `\n\n`), leaving plain tokens unmodified. (4) This stepwise steering reduces reasoning tokens with minimal accuracy loss.

Answering these questions requires a way to probe self-reflection in the model’s internal representations, which prior work has not provided. In this work, we address both questions through the lens of **representation engineering** (Zou et al., 2023). We propose **ReflCtrl**, a framework that identifies a *reflection direction* in the model’s latent space: a direction that separates reflection steps from non-reflection steps in hidden-state activations (see Fig. 1). Specifically, we segment the model’s reasoning trace into steps delimited by natural paragraph breaks, identify reflection-related steps via keywords, and compute the reflection direction as the mean difference in hidden representations between the two sets. Using this direction, we introduce a *stepwise steering method* that injects the direction only at the start of each new reasoning step, enabling fine-grained and efficient control over reflection frequency without degrading generation quality.

Our experiments yield two key findings. First, model activations along the reflection direction are highly predictive of answer correctness, indicating that self-reflection is internally regulated by the model’s uncertainty and providing a mechanistic answer to **RQ1**. Second, reflection is often redundant: applying stepwise steering to suppress unnecessary reflections reduces total reasoning tokens by up to 43.2% while preserving reasoning accuracy, directly addressing **RQ2**. Together, these results offer both a mechanistic account of self-reflection and a practical method for inference-time efficiency.

Our contributions are:

1. We propose **ReflCtrl**, a framework that identifies a reflection direction in LLM latent space and uses it to control self-reflection via stepwise representation steering.
2. We show that activations along the reflection direction correlate with model uncertainty, providing a mechanistic explanation for when and why models initiate self-reflection (**RQ1**).
3. We demonstrate that many reflections are redundant and that suppressing them via stepwise steering reduces inference cost by up to 43.2% without sacrificing reasoning accuracy (**RQ2**).

2 Preliminaries

2.1 Background

Reasoning LLMs are built upon the Transformer decoder architecture (Vaswani et al., 2017). Our method extracts hidden representations from two sub-components of each layer: the self-attention block and the feed-forward MLP block. Each decoder layer l processes the hidden representation $z_l \in \mathbb{R}^d$ as:

$$\begin{aligned}\tilde{z}_l &= z_l + z_l^{\text{attn}}, z_l^{\text{attn}} = \text{Attn}(\text{LN}(z_l)), \\ z_{l+1} &= \tilde{z}_l + z_l^{\text{mlp}}, z_l^{\text{mlp}} = \text{MLP}(\text{LN}(\tilde{z}_l)).\end{aligned}\tag{1}$$

Here, $\text{LN}(\cdot)$ denotes layer normalization, $\text{Attn}(\cdot)$ is the self-attention block, and $\text{MLP}(\cdot)$ is the feed-forward network. We denote $\tilde{z}_l$ as the intermediate state after the attention block, and use z_l^{attn} and z_l^{mlp} as the per-step representations from which we extract the reflection direction.

2.2 Identifying reflection steps

Reasoning LLMs generate extended chain-of-thought traces that interleave forward reasoning with *self-reflection* (DeepSeekAI et al., 2025): “aha moments” at which the model pauses to re-examine, critique, or revise its prior reasoning before proceeding. These self-reflective steps are typically surfaced through characteristic phrases such as “Wait” or “Let me check”, signaling that the model has shifted from producing new reasoning content to evaluating what it has already produced.

To identify self-reflection in model outputs, we employ a two-stage process. First, we segment each reasoning trace into individual steps by leveraging the consistent delimiter pattern used by reasoning LLMs: the token sequence “\n\n” (empty lines). This natural segmentation produces coherent reasoning chunks that serve as our basic unit of analysis. Second, we classify each chunk as either forward reasoning or self-reflection using a keyword-based approach. We search for characteristic phrases that signal the initiation of reflective episodes, such as “Wait” and “Let me check”. This heuristic captures the model’s explicit transitions from generating new reasoning content to evaluating previously generated steps. We validate the reliability of this labeling scheme using an LLM-based judge (Sec. D).

2.3 Empirical motivation for causal control

Self-reflection, while potentially valuable, introduces non-trivial inference overhead. Fig. 2a quantifies this cost: on the DeepSeek-R1-Distill-Llama-8B baseline, reflection steps consume **23.2%** of all reasoning tokens on GSM8k and **35.9%** on MATH-500.

To investigate **RQ2**, we examine how accuracy relates to reflection frequency. Using DeepSeek-R1-Distill-Llama-8B on GSM8k and MATH-500 with $N = 10$ samples per question, we find that correctness rate decreases as reflection rate (# of reflection steps / # of total reasoning steps) increases (Fig. 2b). This may seem surprising at the first glance, as reflection appears to hurt reasoning performance. However, note that the correlation does *not* imply reflection is useless: harder questions usually trigger more reflections. Since those questions have lower accuracy, the reflection frequency is confounded with task difficulty (see Sec. C.4 for deeper analysis).

To disentangle the causal effect of reflection from the confounder of question difficulty, we need to manipulate reflection behavior for the same question. This inspires us to propose **ReflCtrl**: a representation engineering approach that identifies this *reflection direction* from labeled hidden states and steers it at inference time to causally control reflection frequency.

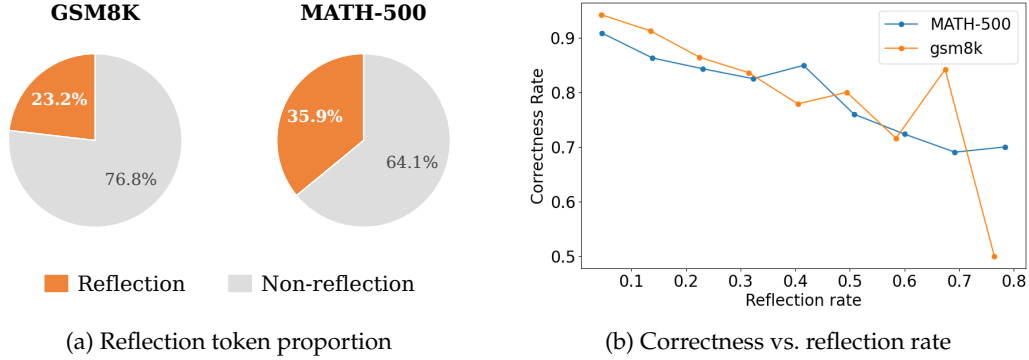


Figure 2: **Reflection overhead and its correlation with accuracy.** *Left:* Reflection steps account for 23.2% and 35.9% of all reasoning tokens on GSM8k and MATH-500, respectively, for DeepSeek-R1-Distill-Llama-8B. *Right:* Higher reflection frequency is associated with lower accuracy on both benchmarks ($N = 10$ samples per question), but this trend is confounded by question difficulty (see Sec. C.4).

3 ReflCtrl: Probing and steering self-reflection

In this section, we propose ReflCtrl, a representation engineering pipeline that enables flexible, inference-time control of the model’s self-reflection without degrading normal generation. The core idea is to identify a *reflection direction*—a direction in the model’s latent space that linearly separates reflection steps from non-reflection steps—and use it both to steer reflection frequency during inference (Sec. 3.2) and to probe the model’s internal state as an approach to **RQ1** (Sec. 4.2). The method proceeds in two stages: computing the reflection direction from labeled hidden states (Sec. 3.1), and applying stepwise representation steering to control self-reflection (Sec. 3.2).

3.1 Extracting the reflection direction

Given the labeled reflection and non-reflection steps from Sec. 2.2, we compute the reflection direction using the mean-difference method from representation engineering (Zou et al., 2023). For each step s at layer l , we extract the hidden representations from the attention and MLP outputs at the *first token* of the step, denoted $z_l^{\{\text{attn}, \text{mlp}\}}(s)$. We choose the first token because it captures the model’s internal state at the moment a new reasoning step is initiated, which is the position where the decision to reflect or continue forward reasoning is made. The reflection direction is then the mean difference between reflection and non-reflection representations:

$$d_l^{\{\text{attn}, \text{mlp}\}} = \frac{1}{|R|} \sum_{s \in R} z_l^{\{\text{attn}, \text{mlp}\}}(s) - \frac{1}{|NR|} \sum_{s \in NR} z_l^{\{\text{attn}, \text{mlp}\}}(s), \quad (2)$$

where R and NR are the sets of reflection and non-reflection steps, respectively. The mean-difference estimator is simple, interpretable, and well-motivated by the linear representation hypothesis (Zou et al., 2023): it produces a single direction that maximally separates the two classes in expectation, without requiring a learned classifier or labeled supervision.

3.2 Stepwise steering

The extracted reflection direction enables inference-time control: adding it to internal representations steers the model toward or away from reflection. The standard additive intervention at layer l modifies the hidden state as:

$$z_{l, \text{intv}}^{\{\text{attn}, \text{mlp}\}} = z_l^{\{\text{attn}, \text{mlp}\}} + \lambda d_l^{\{\text{attn}, \text{mlp}\}}, \quad (3)$$

where $\lambda \in \mathbb{R}$ is the intervention strength: $\lambda < 0$ suppresses reflection and $\lambda > 0$ promotes it.

In prior representation steering work (Zou et al., 2023), this intervention is applied at every token position during autoregressive generation. Let $(x_1, x_2, \dots, x_T)$ denote the generated token sequence. The *all-token* strategy applies Eq. 3 uniformly:

$$z_{l,\text{intv}}^{\{\text{attn},\text{mlp}\}}(x_t) = z_l^{\{\text{attn},\text{mlp}\}}(x_t) + \lambda d_l^{\{\text{attn},\text{mlp}\}}, \quad \forall t \in \{1, \dots, T\}. \quad (4)$$

Because generation is autoregressive, each perturbed hidden state feeds into subsequent attention computations, causing the distributional shift to compound across tokens. At high $|\lambda|$, this cumulative effect pushes the model’s representations far from the training distribution, degrading generation quality and limiting the usable intervention range.

To address this, we propose **stepwise steering**: instead of intervening at every token, we restrict the perturbation to *step-boundary positions*, the first token generated after each reasoning step delimiter. Formally, let $\mathcal{B} = \{t : x_{t-1} \text{ completes the delimiter “\n\n”}\}$ denote the set of step-boundary positions. The stepwise intervention replaces Eq. 4 with:

$$z_{l,\text{intv}}^{\{\text{attn},\text{mlp}\}}(x_t) = z_l^{\{\text{attn},\text{mlp}\}}(x_t) + \lambda \mathbf{1}[t \in \mathcal{B}] d_l^{\{\text{attn},\text{mlp}\}}, \quad (5)$$

where $\mathbf{1}[\cdot]$ is the indicator function. Since the number of reasoning steps $|\mathcal{B}|$ is typically much smaller than the total token count T (e.g., a 2000-token trace may contain only 10–20 steps), stepwise steering drastically reduces the number of perturbed positions.

This design is motivated by two observations. First, the reflection direction is extracted from first-token representations (Sec. 3.1), so intervening at step boundaries targets the positions where the direction is most aligned with the model’s step-initiation mechanism. Second, leaving intermediate tokens within each step unperturbed preserves the model’s natural generation dynamics, avoiding the compounding distributional drift that degrades all-token steering at high $|\lambda|$. As shown in figs. 3a and 3b, stepwise steering maintains the same direction of effect as all-token steering while supporting a substantially wider range of λ , enabling finer-grained control over reflection frequency without sacrificing generation quality.

4 Experiments

In this section, we present experiments evaluating our ReflCtrl framework on three fronts: what triggers self-reflection (RQ1), how reflection influences model reasoning performance (RQ2), and how stepwise steering reduces inference cost in practice.

4.1 Settings

Models. We primarily study the DeepSeek-R1-Distill series (DeepSeekAI et al., 2025), including the Qwen2.5-14B and Llama-8B distilled variants, as these models are publicly available. We also evaluate QwQ-32B (Team, 2024) as a representative non-distilled reasoning model.

Datasets. For math tasks, we use GSM8k (Cobbe et al., 2021) and MATH-500 (Lightman et al., 2023) as benchmarks. For general reasoning, we evaluate on three MMLU (Hendrycks et al., 2020) subsets: Professional Accounting, High School Computer Science, and Formal Logic.

Generation settings. We follow the standard generation configuration for each model. For math tasks, we prompt the model with “Please reason step by step, and put your final answer within `\boxed{\}`”. For MMLU, we use the same prompt and additionally require the final answer to be a single letter. Maximum completion tokens are set to 8192 for all tasks except MATH-500, where we use 16384 due to its higher complexity.

Reflection direction extraction. We extract the reflection direction using GSM8k responses following the procedure in Sec. 3. The final reasoning step is excluded, as it typically consists of a concluding sentence unrelated to the reasoning process.

Steering. Unless otherwise specified, all results use the stepwise steering method proposed in Sec. 3. Interventions are applied to all layers except the first and last six; we study the effect of this choice in Sec. C.8.

4.2 Probing model uncertainty (RQ1)

We first use the reflection direction to address **RQ1**: *when does the model decide to initiate reflection?* We hypothesize that the model’s internal uncertainty is the primary trigger: a reasoning LLM initiates reflection when it is uncertain about the current reasoning trajectory. We conduct two experiments to verify this: If this is correct, the reflection direction should encode information about answer correctness, since uncertainty and eventual correctness are tightly coupled.

Experiment 1: Correctness prediction: If the hypothesis is correct, the reflection direction should encode information about answer correctness, since uncertainty and eventual correctness are tightly coupled. We follow Mielke et al. (2022) and train a logistic regression classifier to predict whether the model’s answer is correct, using features derived from the reflection direction. For each instance, we compute the projection of the hidden representation on the reflection direction at each layer and sub-component, and concatenate these into a feature vector p_{intv} :

$$p_{\text{intv}} = \text{concat}(\{p_l^{\text{attn}}\}_{l=1}^{N_{\text{layer}}}, \{p_l^{\text{mlp}}\}_{l=1}^{N_{\text{layer}}}), \quad (6)$$

where $p_l^{\{\text{attn}, \text{mlp}}\}} = \frac{\langle a_l^{\{\text{attn}, \text{mlp}}\}}, z_l^{\{\text{attn}, \text{mlp}}\}} \rangle}{\|z_l^{\{\text{attn}, \text{mlp}}\}}\|_2}$ is the projection of representation. We extract p_{intv} from the end-of-thinking token (`</think>`) and train the classifier on the GSM8k training set. As a baseline, we train the same classifier using the last-layer embedding of the final token.

Results on the GSM8k test set (table 1) show that reflection-direction features achieve higher AUROC and F1 than the final-layer baseline, despite having lower dimensionality. This suggests that **model uncertainty is encoded in the reflection direction**, and provides evidence that uncertainty is a key mechanistic trigger for self-reflection in reasoning LLMs.

Model	final layer embedding		reflection direction	
	AUROC	F1	AUROC	F1
DS-Llama-8B	0.736	0.946	0.772	0.948
QwQ-32B	0.555	0.636	0.564	0.839
DS-Qwen-14B	0.716	0.929	0.850	0.976

Table 1: **Probing results for uncertainty detection.** We train a logistic regression classifier to predict answer correctness using (i) the last token embedding at the final layer or (ii) feature vector p_{intv} derived from reflection direction. Reflection-based features achieve higher AUROC and F1 scores despite lower dimensionality, suggesting that uncertainty is encoded in the reflection direction.

Experiment 2: Token-level entropy To strengthen this evidence, we compute per-token Shannon entropy (over the full vocabulary) at every generated position for all 1,319 GSM8k questions (DeepSeek-R1-Distill-Llama-8B) and compare the entropy of step i conditioned on whether step $i+1$ is a reflection step. Across 50,456 consecutive step pairs (16,582 pre-reflection, 33,874 pre-non-reflection), steps immediately preceding reflection exhibit significantly higher entropy (table 2) with both mean and max pooling, providing a direct evidence that uncertainty is an important factor in model’s reflection.

4.3 Reflection redundancy (RQ2)

Next, we continue to study **RQ2**: Reflection redundancy. As discussed in Sec. 2, to understand the impact of reflections, we need to intervene on the model to change the number of reflections, and measure the resulting change in accuracy. For this purpose, we apply different intervention strengths and show results in table 3. Two key observations emerge:

Pooling	Pre-Reflect	Pre-Non-Reflect	Cohen’s d
Mean	0.734 ± 0.332	0.599 ± 0.366	0.38
Max	2.401 ± 0.647	2.028 ± 0.789	0.50

Table 2: **Token entropy preceding reflection vs. non-reflection steps.** We compute per-token Shannon entropy at every output position and compare the entropy of step i conditioned on whether step $i+1$ is a reflection step. $n=16,582$ pre-reflection and $n=33,874$ pre-non-reflection step pairs from DS-Llama-8B on GSM8k.

Dataset	Model	Metric	Intervention Strength λ			
			−0.96	−0.48	0	+0.48
GSM8k	DS-Llama-8B	Acc (%) $\uparrow$	88.34 (−1.75pp)	89.46 (−0.63pp)	90.09	90.25 (+0.16pp)
		Tokens $\downarrow$	821.0 (−48.5%)	1032.6 (−35.3%)	1595.7	3577.1 (+124.2%)
	QwQ-32B	Acc (%) $\uparrow$	96.36 (−0.14pp)	96.50 (+0.00pp)	96.50	96.44 (−0.06pp)
		Tokens $\downarrow$	1006.7 (−32.4%)	1162.5 (−21.9%)	1488.6	2256.9 (+51.6%)
	DS-Qwen-14B	Acc (%) $\uparrow$	95.07 (−0.08pp)	95.15 (+0.00pp)	95.15	94.84 (−0.31pp)
		Tokens $\downarrow$	747.8 (−43.2%)	880.2 (−33.1%)	1315.9	3746.4 (+184.7%)
MATH-500	DS-Llama-8B	Acc (%) $\uparrow$	82.14 (−3.84pp)	84.46 (−1.52pp)	85.98	86.90 (+0.92pp)
		Tokens $\downarrow$	2738.1 (−31.6%)	3123.8 (−21.9%)	4000.7	6017.8 (+50.4%)
	QwQ-32B	Acc (%) $\uparrow$	92.72 (−0.34pp)	92.58 (−0.48pp)	93.06	93.08 (+0.02pp)
		Tokens $\downarrow$	2992.9 (−20.9%)	3253.4 (−14.1%)	3786.0	5028.9 (+32.8%)
	DS-Qwen-14B	Acc (%) $\uparrow$	89.22 (−2.22pp)	90.18 (−1.26pp)	91.44	91.86 (+0.42pp)
		Tokens $\downarrow$	2247.1 (−32.2%)	2534.7 (−23.5%)	3315.3	5789.0 (+74.6%)

Table 3: **Accuracy and average reasoning token usage under different intervention strengths.** Results on GSM8k and MATH-500 for three models, averaged over 10 runs. Negative intervention strengths consistently reduce token usage with minimal accuracy loss. Relative changes compared to the baseline ($\lambda = 0$) are shown in parentheses. In the *Tokens* ($\downarrow$) rows, the **bolded** token value (including its relative change) marks the lowest token usage (within each dataset/model pair) among intervention strengths whose *accuracy drop* relative to the baseline is < 1 percentage point (pp). Standard deviations are reported in Sec. C.7.

1. **Reflections are largely redundant in stronger models.** For QwQ-32B, suppressing reflections at strength $\lambda = -0.96$ reduces token usage by 32.4% on GSM8k and 20.9% on MATH-500, with accuracy drops of only 0.14pp and 0.34pp, respectively. Additionally, we note that the redundancy is more significant on larger models. To further isolate the effect of model scale from architecture differences, we evaluate three models from the same family (DeepSeek-R1-Distill-Qwen-1.5B/7B/14B) in Sec. C.5, which further shows larger models are more resilient to the reduction of reflection, implying more redundancy.
2. **Smaller distilled models benefit only marginally from additional reflections.** DeepSeek-R1-Distill-Llama-8B gains only 0.16pp and 0.92pp in accuracy at strength $\lambda = +0.48$, at the cost of approximately 2000 additional tokens per question, which is an unfavorable trade-off in most scenarios.

The full accuracy–token trade-off across all models and datasets is summarized in table 3. Results consistently show that reflection frequency can be substantially reduced with minimal accuracy loss, confirming reflection redundancy across model scales and benchmarks, especially for those larger and stronger models.

Token savings across difficulty levels. A potential concern is whether the token reduction comes primarily from the model “giving up” on hard questions. To further understand the mechanism, we stratify MATH-500 questions into Easy ($>80\%$ baseline accuracy, $n=399$), Medium ($50\text{--}80\%$, $n=41$), and Hard ($<50\%$, $n=60$) based on 10-sample per-question accuracy of DS-Llama-8B (table 4). We have two observations: (1) Token reduction is relatively uniform across all difficulty levels (20–33%), not concentrated in hard questions. Instead,

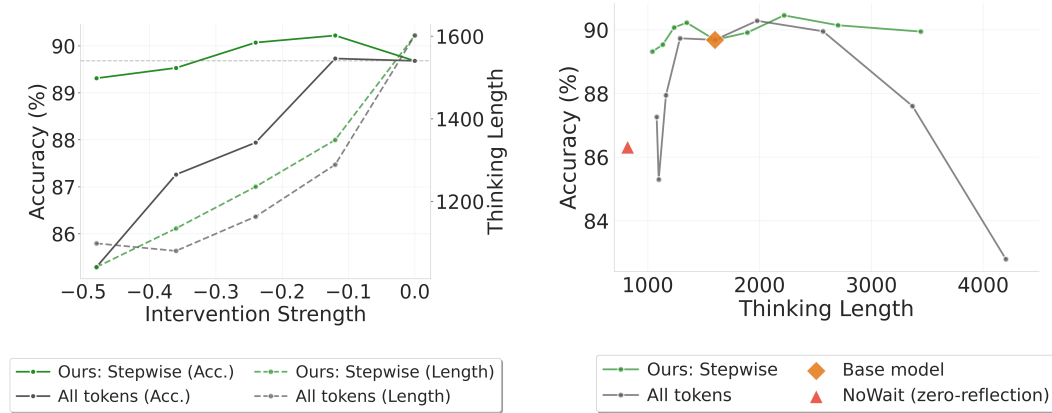
the reduction is *larger* for easy questions (33.2%) than hard ones (20.0%). (2) Regarding the accuracy, hard-question accuracy is unchanged (-0.2pp), likely because the reflection does not help much for those questions.

Difficulty	#Q	$\lambda=0$ Acc	$\lambda=-0.96$ Acc	Token Red.
Easy	399	98.0%	94.8% (-3.2pp)	33.2%
Medium	41	60.0%	53.2% (-6.8pp)	26.3%
Hard	60	12.5%	12.3% (-0.2pp)	20.0%

Table 4: **Per-difficulty stratification on MATH-500** (DS-Llama-8B, $\lambda=-0.96$). Token reduction is relatively uniform across difficulty levels and hard-question accuracy is unchanged.

We further verify that the saved tokens are specifically reflection tokens (not general reasoning): a breakdown analysis shows reflection tokens are reduced $2\text{--}3\times$ more than non-reflection tokens (Sec. C.6).

4.4 Stepwise steering and comparison with baselines



(a) Accuracy and reasoning token usage under different intervention strengths.

(b) Accuracy versus reasoning token usage.

Figure 3: **Comparison of stepwise versus all-token steering.** (a) Accuracy under different intervention strengths when interventions are applied at the start of each reasoning step (stepwise) or at every token (all-token). (b) Accuracy versus reasoning token usage under the two approaches. Stepwise steering preserves accuracy while reducing cost, whereas all-token steering causes significant degradation at high intervention strengths.

We compare the stepwise steering strategy against two baselines: (i) an **all-token** baseline that applies the intervention at every generated token, and (ii) **NoWait** (Wang et al., 2025a), which reduces reflection by directly suppressing reflection-triggering tokens.

As shown in figs. 3a and 3b, stepwise steering offers two clear advantages over all-token intervention on DeepSeek-R1-Distill-Llama-8B (GSM8k):

1. **Stepwise steering avoids accuracy degradation at high intervention strength.** The all-token baseline degrades significantly at strengths below -0.2 or above $+0.3$, while stepwise steering maintains accuracy close to the baseline across the full range.
2. **Stepwise steering achieves higher accuracy at the same token budget.** For any given reasoning token count, stepwise steering consistently outperforms all-token intervention (Fig. 3b).

Compared with NoWait, ReflCtrl provides a continuous performance–cost trade-off via the intervention strength λ , whereas NoWait can only disable reflection entirely. ReflCtrl

Subset	Model	Metric	Intervention Strength λ			
			-0.96	-0.48	0	+0.48
Professional Accounting	DS-Llama-8B	Acc (%) $\uparrow$	50.1 (-6.4pp)	53.4 (-3.1pp)	56.5	57.3 (+0.8pp)
		Tokens $\downarrow$	1453.6 (-30.7%)	1668.8 (-20.4%)	2097.5	2807.7 (+33.9%)
	DS-Qwen-14B	Acc (%) $\uparrow$	78.5 (+0.7pp)	76.8 (-1.0pp)	77.8	77.6 (-0.2pp)
		Tokens $\downarrow$	983.9 (-33.6%)	1103.1 (-25.6%)	1482.1	2470.1 (+66.7%)
	QwQ-32B	Acc (%) $\uparrow$	89.3 (+0.8pp)	89.5 (+1.0pp)	88.5	89.2 (+0.7pp)
		Tokens $\downarrow$	1231.2 (-25.3%)	1313.7 (-20.3%)	1648.0	2234.3 (+35.6%)
HS Computer Science	DS-Llama-8B	Acc (%) $\uparrow$	79.6 (-7.7pp)	82.7 (-4.6pp)	87.3	88.0 (+0.7pp)
		Tokens $\downarrow$	1016.1 (-25.6%)	1157.9 (-15.2%)	1365.4	1970.4 (+44.3%)
	DS-Qwen-14B	Acc (%) $\uparrow$	95.2 (+0.2pp)	95.4 (+0.4pp)	95.0	94.8 (-0.2pp)
		Tokens $\downarrow$	711.9 (-23.7%)	787.9 (-15.6%)	933.5	1498.7 (+60.5%)
	QwQ-32B	Acc (%) $\uparrow$	96.6 (-0.1pp)	96.2 (-0.5pp)	96.7	97.0 (+0.3pp)
		Tokens $\downarrow$	771.6 (-11.4%)	741.9 (-14.8%)	871.0	1004.7 (+15.4%)
Formal Logic	DS-Llama-8B	Acc (%) $\uparrow$	60.5 (-1.6pp)	61.0 (-1.1pp)	62.1	62.7 (+0.6pp)
		Tokens $\downarrow$	2266.5 (-32.9%)	2586.9 (-23.4%)	3378.3	4553.5 (+34.8%)
	DS-Qwen-14B	Acc (%) $\uparrow$	91.8 (-0.8pp)	92.2 (-0.4pp)	92.6	92.8 (+0.2pp)
		Tokens $\downarrow$	1287.2 (-31.9%)	1440.1 (-23.9%)	1891.4	3196.5 (+69.0%)
	QwQ-32B	Acc (%) $\uparrow$	96.3 (+0.6pp)	95.5 (-0.2pp)	95.7	96.0 (+0.3pp)
		Tokens $\downarrow$	1481.4 (-13.7%)	1447.8 (-15.7%)	1716.6	2175.6 (+26.7%)

Table 5: **Accuracy and reasoning token usage under different intervention strengths on MMLU subsets.** Results averaged over 10 runs. Larger models (DS-Qwen-14B and QwQ-32B) are largely insensitive to reflection reduction, saving up to 43.2% of reasoning tokens with negligible accuracy loss. Relative changes compared to the baseline ($\lambda = 0$) are shown in parentheses. In the *Tokens* ($\downarrow$) rows, the **bolded** token value (including its relative change) marks the lowest token usage (within each subset/model pair) among intervention strengths whose *accuracy drop* relative to the baseline is < 1 percentage point (pp).

also achieves smaller accuracy loss at comparable token budgets (see Sec. C.2 for detailed comparisons across models and datasets).

4.5 Generalization to non-mathematical benchmarks

To assess generalization beyond math tasks, we evaluate ReflCtrl on three MMLU subsets. As shown in table 5, the same pattern holds: larger models (DS-Qwen-14B and QwQ-32B) maintain accuracy with substantially fewer reflections, saving up to 33.6% of reasoning tokens, while DS-Llama-8B is more sensitive to reflection reduction. We further validate on HumanEval (Chen et al., 2021), a code generation benchmark, and observe similar trend (Sec. C.3).

5 Related Work

Reasoning LLMs. Recent work has shown that scaling inference-time computation enables LLMs to tackle complex tasks by generating explicit reasoning chains before a final answer (Snell et al., 2024). OpenAI o1 (OpenAI, 2024) pioneered the use of reinforcement learning to develop extended thinking during inference. DeepSeekAI et al. (2025) introduce DeepSeek-R1, a reasoning model trained with GRPO, together with distilled variants that transfer reasoning capability to smaller models. Other open-source reasoning models include QwQ-32B (Team, 2024) and Kimi k1.5 (Kimi Team, 2025). Complementary to training, Muennighoff et al. (2025) show that reasoning length can be controlled at the token level through budget forcing—appending or removing special tokens to extend or truncate thinking. We study DeepSeek-R1-Distill and QwQ-32B as representative open-source models, since access to internal representations is required for our mechanistic analysis.

Self-reflection in reasoning models. A distinctive behavior emerging from training reasoning models is self-reflection: the model’s tendency to spontaneously revisit and revise

prior reasoning steps. DeepSeekAI et al. (2025) first report this phenomenon as the “aha moment” in reinforcement learning, and Yang et al. (2025b) analyze it through the lens of linguistic patterns and uncertainty descriptions. A growing body of work examines whether such reflection is always beneficial: Huang et al. (2024) show that LLMs often fail to self-correct reasoning without external feedback, and Su et al. (2025) find that models systematically overthink easy problems while underthinking hard ones, suggesting that reflection frequency is poorly calibrated to problem difficulty (Sui et al., 2025). Most closely related to our work, Wang et al. (2025a) propose reducing excessive reflection by identifying and suppressing reflection-triggering tokens at inference time. Our approach differs fundamentally: rather than operating on surface-level token patterns, we identify a reflection direction in the model’s latent space and show that it is correlated with the model’s internal uncertainty, providing a mechanistic explanation for when reflection is triggered and a principled handle for controlling it.

Representation engineering on LLMs. Representation engineering (Zou et al., 2023) provides a framework for steering LLM behavior by reading and editing internal activations. Core techniques include Activation Addition (Turner et al., 2023), which computes steering vectors from contrastive prompt pairs, and Inference-Time Intervention (Li et al., 2023), which shifts activations along concept-specific directions in attention heads. Both are grounded in the linear representation hypothesis (Park et al., 2023), which posits that high-level concepts are encoded as linear directions in activation space. These methods have been applied to safety and alignment (Rimsky et al., 2024), explainable decisions (Srivastava et al., 2024; Kulkarni et al., 2025), interpretable concept control (Sun et al., 2025a), and machine unlearning (Li et al., 2025). With the emergence of reasoning models, representation-based methods have been extended to reasoning-specific behaviors: ThinkEdit (Sun et al., 2025b) edits weights of neurons associated with short thinking; RICE (Wang et al., 2025b) adjusts MoE activations to improve reasoning; SEAL (Chen et al., 2025) extracts steering vectors from categorized reasoning steps to calibrate efficiency; and CREST (Zhang et al., 2025) identifies cognitive attention heads and steers them via hidden-state rotation. Our work focuses specifically on **self-reflection** and contributes two elements absent from these methods: a stepwise steering strategy that intervenes only at reasoning step boundaries for fine-grained control without cumulative distribution shift (Sec. 3.2), and a mechanistic analysis linking the reflection direction to model uncertainty (Sec. 4.2).

6 Conclusion

In this work, we propose ReflCtrl, a representation engineering framework for understanding and controlling self-reflection in reasoning LLMs. By identifying a reflection direction in the model’s latent space, ReflCtrl enables direct, inference-time control over reflection frequency. Our stepwise steering strategy, which involves applying interventions only at the start of each new reasoning step, substantially reduces token usage while preserving performance. Across math and general reasoning benchmarks, our experiments establish three findings:

1. Reflection redundancy is common, particularly in stronger models, where reflections can be substantially reduced with minimal accuracy loss.
2. The reflection direction correlates with internal uncertainty signals, suggesting that self-reflection is regulated by the model’s internal uncertainty.
3. Stepwise steering substantially reduces performance loss relative to all-token intervention, cutting total reasoning tokens by up to 43.2% while preserving accuracy.

References

- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. Seal: Steerable reasoning calibration of large language models for free. *arXiv preprint arXiv:2504.07986*, 2025.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- DeepSeekAI, Daming Guo, Dayi Yang, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations*, 2024.
- Kimi Team. Kimi k1.5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*, 2025.
- Akshay Kulkarni, Ge Yan, Chung-En Sun, Tuomas Oikarinen, and Tsui-Wei Weng. Interpretable generative models through post-hoc concept bottlenecks. In *CVPR*, 2025.
- Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. *Advances in Neural Information Processing Systems*, 36:41451–41530, 2023.
- Yongce Li, Chung-En Sun, and Tsui-Wei Weng. Effective skill unlearning through intervention and abstention. *NAACL*, 2025.
- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The twelfth international conference on learning representations*, 2023.
- Sabrina J. Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. Reducing conversational agents’ overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 10:857–872, 2022. doi: 10.1162/tacl_a_00494. URL <https://aclanthology.org/2022.tacl-1.50/>.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.
- OpenAI. Openai o1 system card, 2024.
- Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. *arXiv preprint arXiv:2311.03658*, 2023.
- Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, 2024.

- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.
- Divyansh Srivastava, Ge Yan, and Tsui-Wei Weng. Vlg-cbm: Training concept bottleneck models with vision-language guidance. *NeurIPS*, 2024.
- Jinyan Su, Jennifer Healey, Preslav Nakov, and Claire Cardie. Between underthinking and overthinking: An empirical study of reasoning length and correctness in llms. *arXiv preprint arXiv:2505.00127*, 2025.
- Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, et al. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*, 2025.
- Chung-En Sun, Tuomas Oikarinen, Berk Ustun, and Tsui-Wei Weng. Concept bottleneck large language models. In *ICLR*, 2025a.
- Chung-En Sun, Ge Yan, and Tsui-Wei Weng. Thinkedit: Interpretable weight editing to mitigate overly short thinking in reasoning models. *EMNLP*, 2025b.
- Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown, November 2024. URL <https://qwenlm.github.io/blog/qwq-32b-preview/>.
- Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. *arXiv preprint arXiv:2308.10248*, 2023.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Chenlong Wang, Yuanning Feng, Dongping Chen, Zhaoyang Chu, Ranjay Krishna, and Tianyi Zhou. Wait, we don't need to "wait"! removing thinking tokens improves reasoning efficiency. *arXiv preprint arXiv:2506.08343*, 2025a.
- Mengru Wang, Xingyu Chen, Yue Wang, Zhiwei He, Jiahao Xu, Tian Liang, Qiuzhi Liu, Yunzhi Yao, Wenxuan Wang, Ruotian Ma, et al. Two experts are all you need for steering thinking: Reinforcing cognitive effort in moe reasoning models without additional training. *arXiv preprint arXiv:2505.14681*, 2025b.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Wang, Bowen Zhu, Chengyuan Chen, Dayiheng Li, Fei Jiang, Hao Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.
- Shu Yang, Junchao Wu, Xin Chen, Yunze Xiao, Xinyi Yang, Derek F Wong, and Di Wang. Understanding aha moments: from external observations to internal mechanisms. *arXiv preprint arXiv:2504.02956*, 2025b.
- Zhenyu Zhang, Xiaoxia Wu, Zhongzhu Zhou, Qingyang Wu, Yineng Zhang, Pragaash Ponnusamy, Harikaran Subbaraj, Jue Wang, Shuaiwen Leon Song, and Ben Athiwaratkun. Understanding and steering the cognitive behaviors of reasoning models at test-time. *arXiv preprint arXiv:2512.24574*, 2025.
- Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, et al. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

Appendix

A Reasoning sample of LLM	14
B Reflection direction attribution to attention heads	15
C Additional empirical results	16
C.1 Effect of intervention strength on reflection count	16
C.2 Comparison with NoWait	16
C.3 Generalization to code generation	16
C.4 Study of model performance on reflection rate	16
C.5 Same-family model scale comparison	18
C.6 Token breakdown: reflection vs. non-reflection	19
C.7 Statistical reporting: standard deviations	19
C.8 Ablation study: impact of intervention layers	20
C.9 Ablation study: impact of components	20
D Validation of the reflection keyword heuristic	22
E Computational resources	24
F LLM Disclosure	24

A Reasoning sample of LLM

Fig. 4 shows an example of an LLM reasoning process. From the figure, we can see:

1. The model’s reasoning is organized as steps separated by “\n\n”.
2. The reflection process, marked in red, is triggered by specific keywords such as “wait” in this example.

This justifies our reflection direction extraction and intervention method.

Question:

Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers’ market daily for \$2 per fresh duck egg. How much in dollars does she make every day at the farmers’ market?

Thinking Process:

Okay, let me try to figure out how much Janet makes every day at the farmers’ market. So, the problem says that Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers’ market daily for \$2 per fresh duck egg. I need to find out how much she makes each day. Hmm, let me break this down step by step.

First, let me figure out how many eggs Janet starts with each day. The problem says she lays 16 eggs per day. So, she begins with 16 eggs. Then she eats three for breakfast every morning. So, she uses 3 eggs for breakfast. After that, she bakes muffins for her friends, using four eggs each day. So, she uses 4 eggs for baking.

Wait, so she uses 3 eggs in the morning and 4 eggs for the muffins. Let me calculate the total number of eggs she uses in a day. So, 3 plus 4 is 7 eggs. So, she uses 7 eggs each day.

She starts with 16 eggs, uses 7, so how many eggs does she have left? Let me subtract 7 from 16. 16 minus 7 is 9. So, she has 9 eggs left after using them for breakfast and baking.

Now, these remaining 9 eggs, she sells at the farmers’ market. The problem says she sells each fresh duck egg for \$2. So, I need to figure out how much money she makes from selling these 9 eggs.

To find the total revenue, I can multiply the number of eggs she sells by the price per egg. So, 9 eggs times \$2 per egg. Let me compute that. 9 times 2 is 18. So, she makes \$18 each day at the farmers’ market.

Wait, let me double-check my steps to make sure I didn’t make a mistake. She lays 16 eggs, uses 3 for breakfast, 4 for muffins, which totals 7 eggs used. 16 minus 7 is 9 eggs left. 9 eggs times \$2 is \$18. Hmm, that seems right.

I think that makes sense. So, she starts with 16, uses 7, sells 9, making her \$18 each day. Yeah, that adds up.

Just to visualize it, maybe I can represent it with some equations. Let me define variables for clarity. Let E be the number of eggs laid per day, which is 16. Let B be the number of eggs she eats for breakfast, which is 3. Let M be the number of eggs she uses for muffins, which is 4. So, the total eggs used per day would be B plus M, so 3 plus 4 is 7.

Then, the eggs she has left would be E minus (B + M), which is 16 minus 7, which is 9. Then, the revenue R from selling these remaining eggs is 9 times the price per egg, which is 2. So, R is 9 times 2, which is 18.

Yes, that all checks out. So, I think I did it correctly. She makes \$18 each day at the farmers’ market.

Final Answer Janet makes \boxed{18} dollars every day at the farmers’ market.

Figure 4: Sample reasoning process of DeepSeek-R1-Distill-Llama-8B on a GSM8k question.

B Reflection direction attribution to attention heads

In this section, we study the attribution of the reflection direction to individual attention heads. We take the reflection direction at self-attention layers from a DeepSeek-R1-Distill-Qwen-1.5B model, compute the projection of the average head activation onto the reflection direction, and plot the heatmap in Fig. 5. From the figure, we observe:

1. The last layer (layer 27) has the largest projection magnitude among all layers, likely because this layer directly controls the generation of keywords that trigger reflection (e.g., “wait”).
2. Attention heads with positive projection onto the reflection direction are sparse and concentrated in deeper layers, suggesting that these heads may directly control the model’s reflection behavior.

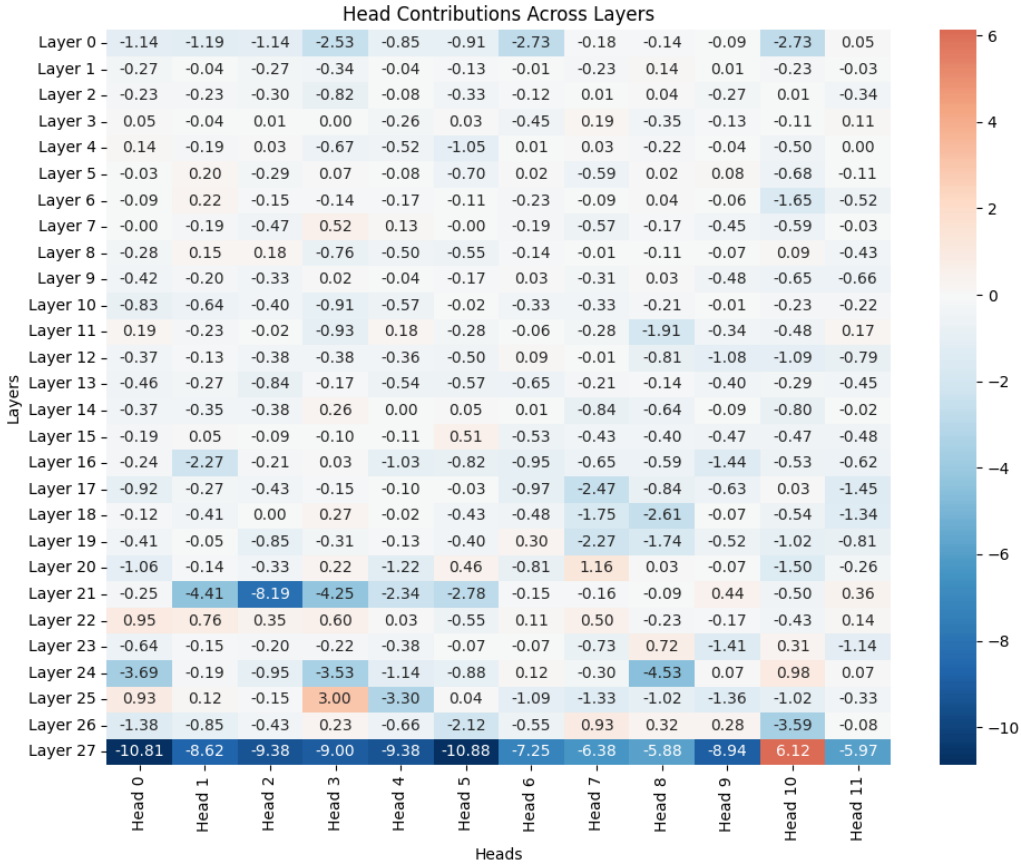


Figure 5: Projection of average attention head activations onto the reflection direction. Results use the GSM8k dataset and a DeepSeek-R1-Distill-Qwen-1.5B model (28 layers, 12 attention heads per layer).

C Additional empirical results

C.1 Effect of intervention strength on reflection count

Fig. 6 shows the accuracy and number of reflection steps under different intervention strengths for DeepSeek-R1-Distill-Llama-8B (distilled) and QwQ-32B (non-distilled) on GSM8k. Accuracy remains largely stable across the intervention range, while the number of reflection steps decreases monotonically as intervention strength becomes more negative, confirming that the extracted reflection direction causally controls reflection frequency.

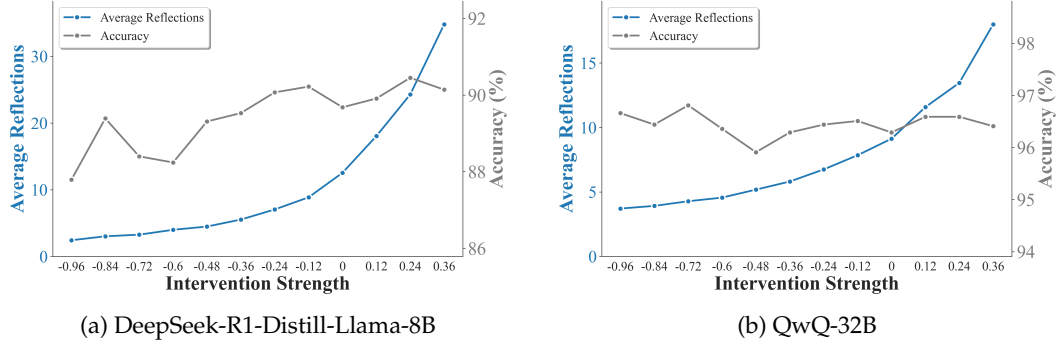


Figure 6: **Accuracy and number of reflection steps under different intervention strengths.** Results are shown for DeepSeek-R1-Distill-Llama-8B (distilled model) and QwQ-32B (non-distilled model) on GSM8k. Accuracy remains largely stable, while the number of reflection steps decreases as intervention strength decreases.

C.2 Comparison with NoWait

Fig. 7 compares the accuracy–token trade-off of ReflCtrl against NoWait (Wang et al., 2025a), which reduces reflection by directly suppressing reflection-triggering tokens. ReflCtrl provides a continuous trade-off via the intervention strength λ , whereas NoWait can only disable reflection entirely. Across both models and datasets, ReflCtrl achieves smaller accuracy loss at comparable token budgets.

C.3 Generalization to code generation

To evaluate whether ReflCtrl generalizes beyond mathematical and knowledge-intensive tasks, we apply it to HumanEval (Chen et al., 2021), a benchmark of 164 Python programming problems that tests functional code generation. Each problem provides a function signature and docstring; the model must produce a correct implementation. We report pass@1 averaged over 3 samples per problem. As shown in table 6, negative intervention strengths reduce reasoning tokens by up to 24% while maintaining pass@1 with a performance drop no more than 2pp compared to the baseline across all models, confirming that steering has negligible impact on code generation and that the method is safe to apply beyond mathematical reasoning tasks.

C.4 Study of model performance on reflection rate

We further study how model performance depends on reflection rate by partitioning questions into three difficulty classes based on per-question accuracy: Hard ($acc < 50\%$), Medium ($50\% \leq acc \leq 80\%$), and Easy ($acc > 80\%$). We report correctness vs. reflection rate for each class under the base model and under intervention $\lambda = -0.96$ in Fig. 8. Three observations emerge:

1. Harder questions trigger more reflection: easy questions have an average reflection rate of 25.8% versus 37.5% for hard questions.

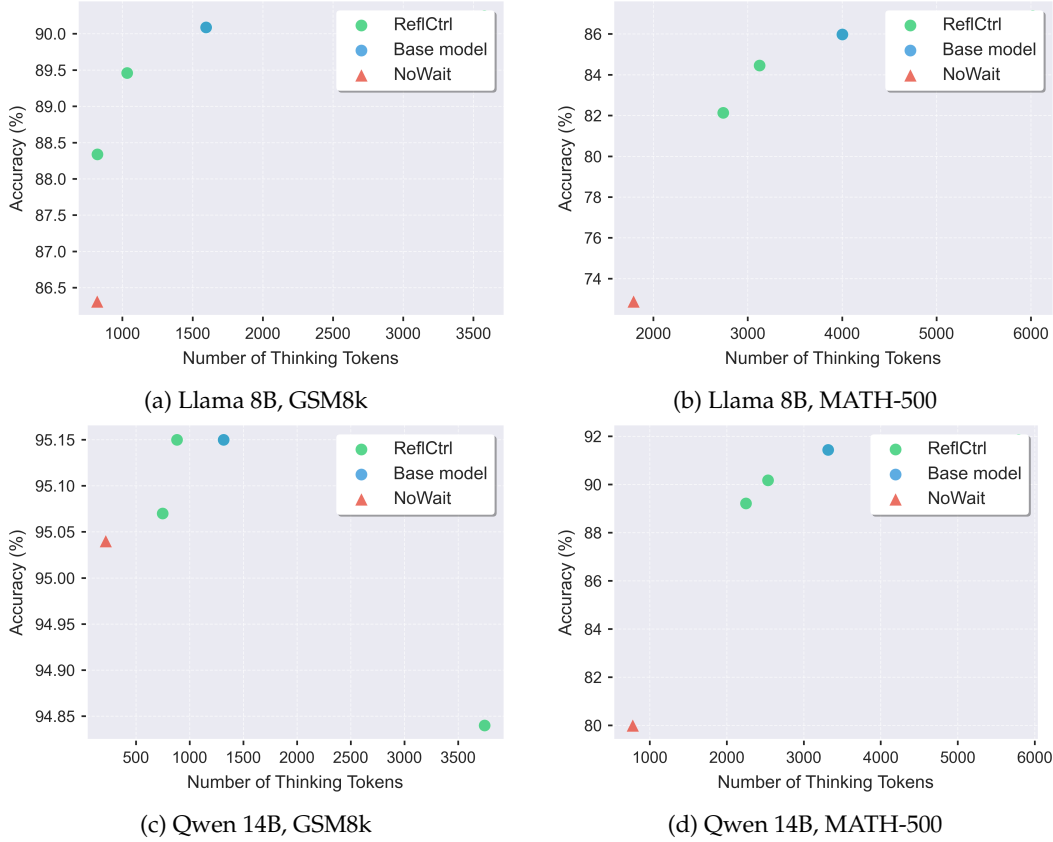
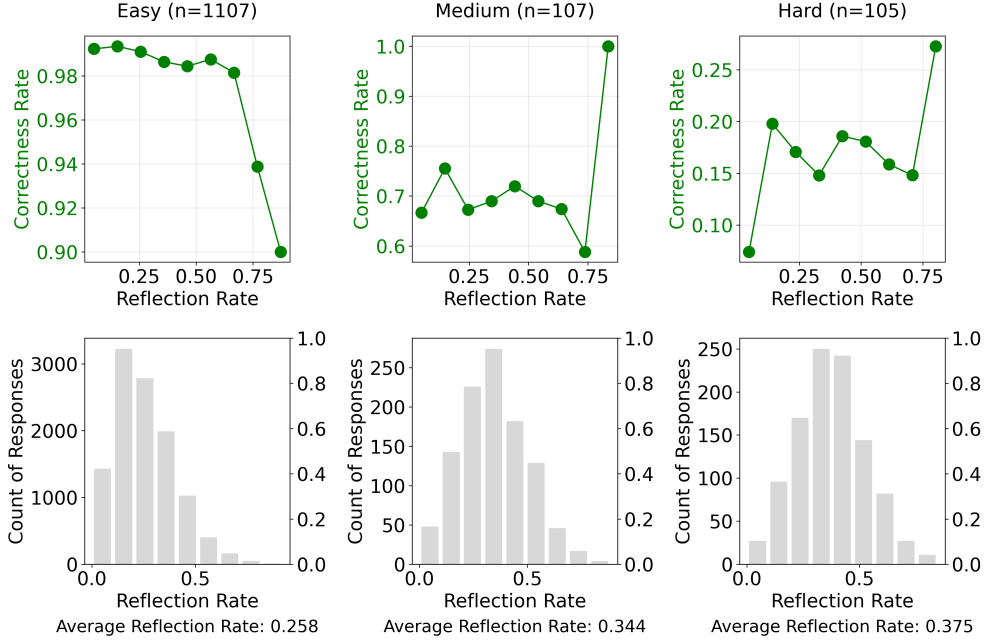


Figure 7: **Accuracy versus reasoning token usage for ReflCtrl compared with NoWait (Wang et al., 2025a).** Results are shown for DeepSeek-R1-Distill-Llama-8B and DeepSeek-R1-Distill-Qwen-14B across GSM8k and MATH-500 benchmarks. ReflCtrl allows fine-grained control over the trade-off between accuracy and reasoning cost via intervention strength, while NoWait can only suppress reflections entirely. Additionally, ReflCtrl achieves lower performance loss for similar token usage.

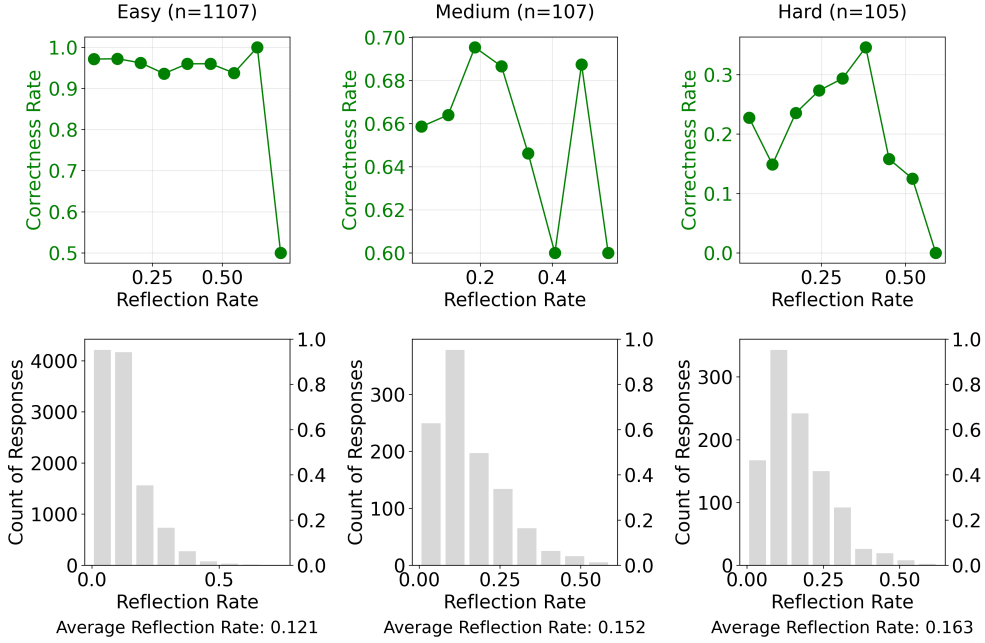
Dataset	Model	Metric	Intervention Strength λ			
			-0.96	-0.48	0	+0.48
HumanEval	DS-Llama-8B	pass@1 (%) $\uparrow$	85.77 (+1.42pp)	87.20 (+2.85pp)	84.35	85.57 (+1.22pp)
		Tokens $\downarrow$	2074.0 (-22.0%)	2269.0 (-14.6%)	2658.0	3353.0 (+26.1%)
	DS-Qwen-14B	pass@1 (%) $\uparrow$	95.12 (+0.00pp)	95.33 (+0.21pp)	95.12	95.33 (+0.21pp)
		Tokens $\downarrow$	1667.0 (-24.3%)	1776.0 (-19.3%)	2201.0	3576.0 (+62.5%)
	QwQ-32B	pass@1 (%) $\uparrow$	98.17 (+0.81pp)	96.95 (-0.41pp)	97.36	95.93 (-1.43pp)
		Tokens $\downarrow$	1945.0 (-19.6%)	2044.0 (-15.5%)	2418.0	3183.0 (+31.6%)

Table 6: **Pass@1 and average reasoning token usage under different intervention strengths on HumanEval.** Results averaged over 3 samples per problem (164 problems). Relative changes compared to the baseline ($\lambda = 0$) are shown in parentheses. In the *Tokens* ($\downarrow$) rows, the **bolded** token value (including its relative change) marks the lowest token usage (within each model) among intervention strengths whose *accuracy drop* relative to the baseline is < 1 percentage point (pp).

2. Within each difficulty class, correctness rate is not correlated with reflection rate, suggesting that model reflections may be redundant even on difficult questions.
3. After intervention, reflection rates decrease across all difficulty classes, but harder questions still receive more reflections than easier ones.



(a) Base model. Top: Correctness rate vs. reflection rate. Bottom: Reflection rate distribution.



(b) With intervention. Top: Correctness rate vs. reflection rate. Bottom: Reflection rate distribution.

Figure 8: Correctness rate and reflection rate across three question difficulty classes for DeepSeek-R1-Distill-Llama-8B on GSM8k.

C.5 Same-family model scale comparison

To isolate the effect of model scale from architecture differences, we evaluate three models from the same family and distillation procedure: DeepSeek-R1-Distill-Qwen-1.5B, 7B, and 14B. All three are distilled from DeepSeek-R1 using identical training protocols, varying only

in model capacity. We extract a separate reflection direction for each model from GSM8k traces and apply steering at $\lambda \in \{0, -0.48, -0.96\}$.

Reflection rates. table 7 shows that reflection rates are similar across scales (19–23%), indicating that the propensity to reflect is not strongly scale-dependent.

Model	Total Steps	Reflect Steps	Reflect Rate
DS-Qwen-1.5B	25,508	5,981	23.4%
DS-Qwen-7B	46,715	10,722	23.0%
DS-Qwen-14B	42,794	8,378	19.6%

Table 7: Reflection rates on GSM8k across the DeepSeek-R1-Distill-Qwen family (keyword heuristic).

Steering results. As shown in table 8, larger models tolerate stronger reflection suppression with dramatically less accuracy cost. The efficiency ratio (token reduction per percentage point of accuracy drop) increases monotonically from $39\times$ (1.5B) to $53\times$ (7B) to $540\times$ (14B), confirming that reflection redundancy scales with model capacity within a single architecture family.

Model	λ	Accuracy	Tokens	Δ Acc	Δ Tok
DS-Qwen-1.5B	0.0	85.90%	1979	—	—
	−0.48	84.89%	918	−1.01pp	−53.6%
	−0.96	84.33%	781	−1.57pp	−60.5%
DS-Qwen-7B	0.0	92.90%	1517	—	—
	−0.48	92.67%	992	−0.23pp	−34.6%
	−0.96	92.01%	806	−0.89pp	−46.9%
DS-Qwen-14B	0.0	95.15%	1316	—	—
	−0.48	95.15%	880	+0.00pp	−33.1%
	−0.96	95.07%	748	−0.08pp	−43.2%

Table 8: **Same-family scale comparison on GSM8k.** Three DeepSeek-R1-Distill-Qwen models (1.5B/7B/14B), distilled with the same procedure, evaluated under identical steering settings. Larger models achieve similar token reduction with dramatically less accuracy cost, confirming that reflection redundancy scales with model capacity.

C.6 Token breakdown: reflection vs. non-reflection

To confirm that steering specifically targets reflection tokens rather than reducing reasoning output indiscriminately, we decompose the total token count into reflection and non-reflection components. We tokenize all reasoning traces using the model’s own tokenizer, split into steps at paragraph boundaries, classify each step as reflection or non-reflection via keyword matching, and count tokens per category. Steps containing the final-answer phrase are excluded from both categories, consistent with the direction extraction procedure (Sec. 3); the Reflect % column is thus computed as reflection tokens divided by the sum of reflection and non-reflection tokens.

table 9 shows results for DS-Qwen-1.5B and DS-Qwen-7B on GSM8k. At $\lambda = -0.96$, reflection tokens are reduced by 75–85% while non-reflection tokens are reduced by only 34–44%—a 2–3 $\times$ disproportionate reduction. The reflection-token fraction drops from 33–40% of reasoning tokens to approximately 15%, confirming that the intervention specifically suppresses reflection behavior rather than shortening reasoning uniformly.

C.7 Statistical reporting: standard deviations

tables 10 and 11 report the standard deviations across 10 independent runs for all entries in the main results tables. Accuracy standard deviations range from 0.20–1.18pp, confirming that the reported differences are well above run-to-run noise.

Model	λ	Reflect Tok	Non-Reflect Tok	Reflect %
DS-Qwen-1.5B	0.0	775	1141	40.4%
	-0.48	189 (-75.6%)	699 (-38.7%)	21.3%
	-0.96	113 (-85.4%)	636 (-44.3%)	15.1%
DS-Qwen-7B	0.0	492	997	33.0%
	-0.48	185 (-62.3%)	780 (-21.8%)	19.2%
	-0.96	120 (-75.5%)	660 (-33.8%)	15.4%

Table 9: **Reflection vs. non-reflection token breakdown on GSM8k.** Reflection tokens are reduced 2–3 $\times$ more than non-reflection tokens across both models and steering strengths, confirming that the intervention specifically targets reflection behavior.

Model	λ	Accuracy Std (pp)	Tokens Std
DS-Llama-8B	-0.96	0.73	11
	-0.48	0.46	15
	0	0.46	45
	+0.48	0.55	61
QwQ-32B	-0.96	0.22	13
	-0.48	0.22	12
	0	0.25	12
	+0.48	0.27	32
DS-Qwen-14B	-0.96	0.20	7
	-0.48	0.37	11
	0	0.25	15
	+0.48	0.32	74

Table 10: **Standard deviations for GSM8k results** (10 independent runs per setting).

Model	λ	Accuracy Std (pp)	Tokens Std
DS-Llama-8B	-0.96	0.92	115
	-0.48	0.95	73
	0	1.18	105
	+0.48	0.51	132
QwQ-32B	-0.96	0.47	55
	-0.48	0.64	42
	0	0.37	41
	+0.48	0.32	74
DS-Qwen-14B	-0.96	0.71	46
	-0.48	0.60	66
	0	0.44	81
	+0.48	0.63	113

Table 11: **Standard deviations for MATH-500 results** (10 independent runs per setting).

C.8 Ablation study: impact of intervention layers

We ablate the choice of which layers receive the intervention, testing two strategies: skipping the first k layers and skipping the last k layers. As shown in Fig. 9, accuracy is highest when skipping both the first and last six layers, which we adopt as the default configuration.

C.9 Ablation study: impact of components

We ablate which model component receives the steering intervention: self-attention only, MLP only, or both (default). All experiments use DeepSeek-R1-Distill-Llama-8B on GSM8k with 3 samples per question. Fig. 10 shows accuracy and token usage at strengths $\lambda \in \{0, -0.5, -1.0\}$. We find that **combining both components yields the strongest token**

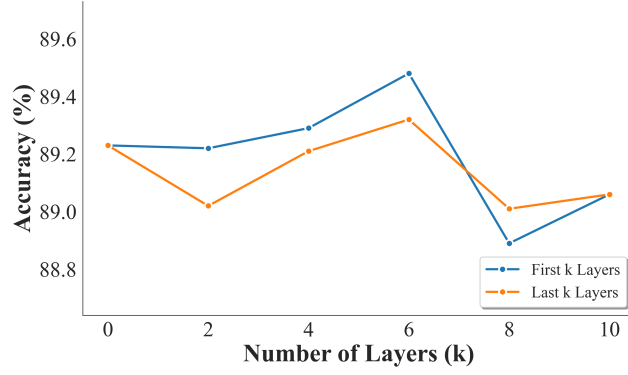


Figure 9: **Effect of applying interventions to different layers.** We vary the number of skipped layers at the bottom and top of the network at fixed strength $\lambda = -0.48$. Accuracy is highest when skipping the first and last six layers, which is adopted as the default.

reduction. At $\lambda = -1.0$, the default setting reduces tokens to 830, compared to 1,161 for attention-only and 925 for MLP-only, while maintaining comparable accuracy.

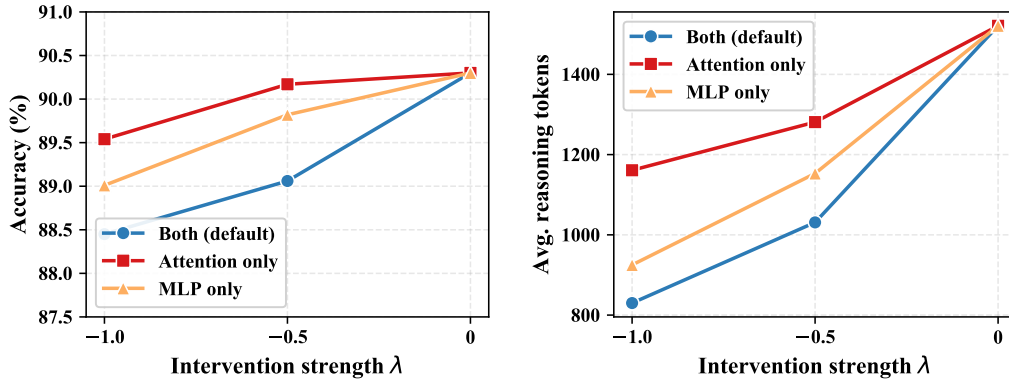


Figure 10: Component ablation: accuracy and reasoning token usage when steering is applied to self-attention only, MLP only, or both (default). Results on GSM8k with DeepSeek-R1-Distill-Llama-8B.

Table 12: Reflection heuristic validation against an LLM judge (Qwen3-32B) on DeepSeek-R1-Distill-Llama-8B. Precision and recall treat the judge labels as reference.

Labeled steps	28,413
Chunks evaluated	9,306
Heuristic pos. rate	15.8%
Judge pos. rate	17.3%
TP / FP / FN / TN	2,899 / 1,577 / 2,018 / 21,919
Agreement	87.3%
Precision	64.8%
Recall	59.0%
F_1	61.7%

D Validation of the reflection keyword heuristic

The reflection direction extraction (Sec. 2.2) relies on a keyword-based heuristic that labels a reasoning step as *reflective* if it contains any of the keywords: wait, let me check, double-check, or alternatively (case-insensitive). To assess the quality of this heuristic, we compare its labels against an independent LLM judge on unsteered ($\lambda = 0$) GSM8k reasoning traces.

Setup. We use Qwen3-32B (Yang et al., 2025a) as the judge, served via vLLM with 4-way tensor parallelism and greedy decoding (temperature = 0). Each reasoning trace is segmented into steps by splitting on the “\n\n” delimiter. Steps are grouped into non-overlapping chunks of $k=4$ target steps, each preceded by $c=2$ context steps from the prior chunk for local coherence. We audit 1,000 DeepSeek-R1-Distill-Llama-8B reasoning traces.

The judge is instructed to label whether each target step is the *start* of a new reflection span—directly mirroring the onset signal used by our stepwise steering intervention. The full system prompt is shown below:

You are an expert annotator for mathematical reasoning traces. Your task is to identify whether each TARGET step is the START of a self-reflection span.

Definition of self-reflection start step: Label 1 only if the step is the first step where reflection behavior begins (e.g., doubt/check/correction/alternative-path trigger starts at this step).

Label 0 for: continuation steps inside an already-started reflection span, normal forward derivation, routine arithmetic/logical progress, final answer formatting without re-checking intent.

Important rules: Only label TARGET steps, not CONTEXT steps. Use CONTEXT to decide whether reflection already started before a TARGET step. If reflection is already ongoing in CONTEXT, early TARGET continuation steps must be 0.

The judge returns a structured JSON response with a binary reflection label, confidence score, and brief rationale for each target step.

Results. table 12 summarizes the audit over 28,413 labeled steps across 9,306 chunks. The overall agreement between the heuristic and the judge is 87.3%, driven by the large number of non-reflective steps on which both methods concur. The per-class metrics show that the keyword heuristic achieves 64.8% precision but only 59.0% recall, indicating that the heuristic is a conservative detector—most steps it flags are genuine reflection onsets, but it misses a substantial fraction that the judge identifies.

Inspection of disagreement cases reveals two systematic patterns:

1. **False negatives** (heuristic misses) arise when the model initiates reflection with phrases not covered by the keyword list, such as “*Hold on, is that right?*” or implicit re-checking without a lexical trigger.
2. **False positives** (heuristic over-detections) occur when a keyword appears inside an already-active reflection span; the heuristic flags every keyword occurrence, whereas the judge correctly restricts the positive label to the span’s first step.

Despite the heuristic’s imperfect recall, the extracted steering directions remain effective (table 3). We attribute this to the contrastive nature of the mean-difference extraction: even a noisy partition of steps into reflective vs. non-reflective sets yields a meaningful direction, because the true positives dominate the reflective centroid while the missed steps only slightly dilute the non-reflective centroid.

Direction robustness: keyword vs. LLM-judge labels. To directly quantify the impact of label noise on the extracted direction, we re-extract the reflection direction using the LLM judge’s labels instead of the keyword heuristic. Using all 1,319 GSM8k traces from DeepSeek-R1-Distill-Llama-8B (52,346 total steps, of which 8,427 are labeled by the judge), we compute the mean-difference direction from the judge-labeled reflective vs. non-reflective partitions and compare it with the keyword-based direction used in all main experiments.

table 13 reports the per-component cosine similarity between the two directions across all 64 sub-components ($32 \text{ layers} \times \{\text{attention, MLP}\}$). The mean cosine similarity is 0.81, indicating that both labeling methods recover largely the same linear direction.

	Mean	Std	Min	Max
Cosine similarity	0.81	0.045	0.71	0.94

Table 13: Cosine similarity between keyword-based and LLM-judge-based reflection directions across all 64 sub-components (DS-Llama-8B, GSM8k).

To confirm this translates to equivalent downstream performance, we run steering with the LLM-judge direction at $\lambda = -0.48$ on both GSM8k and MATH-500 (table 14). The two directions produce nearly identical results: accuracy differences are within 1.6pp and token counts are comparable, confirming that the mean-difference extraction is inherently robust to label noise.

Dataset	Direction source	Accuracy	Tokens
GSM8k	Keyword (paper)	89.46%	1033
	LLM-judge	88.63%	1041
MATH-500	Keyword (paper)	84.46%	3124
	LLM-judge	82.87%	2765

Table 14: Steering comparison using keyword-based vs. LLM-judge-based reflection directions (DS-Llama-8B, $\lambda = -0.48$). Both directions produce equivalent steering outcomes, confirming robustness to label noise in direction extraction.

E Computational resources

The main experiments were conducted on 8 NVIDIA RTX 6000 Ada GPUs for approximately 200 hours.

F LLM Disclosure

This work uses Qwen3-32B ([Yang et al., 2025a](#)) as an LLM-based judge to validate the reflection keyword heuristic (Sec. D), as well as the DeepSeek-R1-Distill family and QwQ-32B in the main experiments. The writing of this paper was polished with the assistance of an LLM.