Unveiling the Latent Directions of Reflection in Large Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Reflection, the ability of large language models (LLMs) to evaluate and revise their own reasoning, has been widely used to improve performance on complex reasoning tasks. Yet, most prior work emphasizes designing reflective prompting strategies or reinforcement learning objectives, leaving the inner mechanisms of reflection underexplored. In this paper, we investigate reflection through the lens of latent directions in model activations. We propose a methodology based on activation steering to characterize how instructions with different reflective intentions: no reflection, intrinsic reflection, and triggered reflection. By constructing steering vectors between these reflection levels, we demonstrate that (1) new reflection-inducing instructions can be systematically identified, (2) reflective behavior can be directly enhanced or suppressed through activation interventions, and (3) suppressing reflection is considerably easier than stimulating it. Experiments on GSM8k-adv with Qwen2.5-3B and Gemma3-4B reveal clear stratification across reflection levels, and steering interventions confirm the controllability of reflection. Our findings highlight both opportunities (e.g., reflection-enhancing defenses) and risks (e.g., adversarial inhibition of reflection in jailbreak attacks). This work opens a path toward mechanistic understanding of reflective reasoning in LLMs.

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities in multi-step reasoning tasks [1, 2, 3, 4], with reflection playing a central role in their success [5, 6, 7, 8]. Reflection enables a model to reconsider its reasoning process, identify errors, and revise its conclusions, thereby producing more reliable outputs [5, 8]. While reflection has been operationalized in diverse ways, such as multi-agent frameworks [6, 7], long chain-of-thought prompting [9, 10, 11], and iterative refinement [8, 12, 13], the mechanisms underlying how reflection emerges in LLMs remain poorly understood. Most prior research has treated reflection as a behavioral property to be exploited, rather than as a latent phenomenon to be explained.

In this paper, we move beyond behavioral prompting strategies and instead focus on the mechanistic interpretability of reflection. Building on recent advances in activation steering [14, 15, 16], we investigate whether reflection aligns latent directions in a model's hidden space. Our contributions are summarized as follows:

- We categorize reflection into three levels: No Reflection, Intrinsic Reflection, and Triggered Reflection. This stratification enables the construction of steering vectors that capture the latent transitions between different reflective states.
- Using these steering vectors, we demonstrate a principled approach to discovering new reflection-inducing prompts, moving beyond trial-and-error prompt design.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

- We show that reflective behavior can be directly modulated through activation steering, enabling both enhancement and inhibition of reflection at inference time.
- Our findings reveal an asymmetry: suppressing reflection is easier than inducing it. This observation not only sheds light on the underlying mechanisms of reasoning but also raises potential security concerns, as malicious actors could exploit reflection inhibition to bypass model safeguards.

Table 1: List of Instruction for Reflection in Related Works

Words	Source
wait	[5]
wait	[17]
wait, alternatively, double-check, make sure, another way, verify, to confirm	[18]
wait, alternatively	[19]
wait, alternatively, recheck, retry, however	[20]
wait, alternatively, double-check, let me check, emm, hmm	[10]

2 Methodology

2.1 Problem Formulation

We begin by defining the type of **Reflection** considered in this work. Specifically, we focus on **Situational Reflection** [5], where a model reflects on reasoning generated by another source (e.g., a different model). Other forms, such as **Self-reflection**—where a model critiques its own outputs—are beyond our scope. We choose situational reflection because it provides a more controlled setting to study how models correct deliberately induced errors. Importantly, in our formulation the errors are introduced within the *reasoning steps* rather than only in the final answer. This contrasts with works like [21], which focus on correcting end outputs in non-reasoning tasks. An illustrative example, adapted from [5], is shown in Figure 1: the prompt presents a GSM8k math problem, followed by a deliberately flawed chain-of-thought, and ends with an instruction to trigger reflection (e.g., *wait*, *alternatively*). Prior work has proposed various trigger instructions (summarized in Table 1), but their selection has largely based on intuition rather than systematic analysis. This raises two key research questions:

- How can we systematically identify effective trigger instructions, rather than relying on trial-and-error?
- Do effective trigger instructions correspond to latent directions in the hidden space that implicitly induce the self-reflection process?

To address these questions, we propose a methodology grounded in *activation steering*. Specifically, we first categorize reflection into three levels—*No Reflection*, *Intrinsic Reflection*, and *Triggered Reflection*—to establish a structured framework for analysis. We then compute *steering vectors* between these levels, capturing the latent directions that separate different reflective behaviors. These steering vectors serve two purposes:

- They allow us to discover new instructions beyond those reported in prior work by comparing the alignment of candidate tokens with the reflection-related steering direction.
- They enable controlled interventions, where reflection can be enhanced or inhibited by adding or subtracting steering vectors at selected layers.

2.2 Three Levels Reflection

In this section, we formalize the different levels of reflection. Unlike prior works (e.g., [5, 18]) which explicitly give instructions to trigger the act of reflection, we emphasize that reflection can occur even without explicit triggers: LLMs sometimes spontaneously revise their reasoning, a behavior we call *intrinsic reflection*. This usually perform worse than explicitly triggered reflection. Conversely, reflection can also be suppressed entirely by instructing the model to output an answer directly after

Prompt: Answer the question: John drives for 3 hours at a speed of 60 mph and then turns around because he realizes he forgot something very important at home. He tries to get home in 4 hours but spends the first 2 hours in standstill traffic. He spends the next half-hour driving at a speed of 30mph, before being able to drive the remaining time of the 4 hours going at 80 mph. How far is he from home at the end of those 4 hours? Please always end your response with the final numerical answer

Let's solve this step by step ...

When he turned around he was 3*60=«3*60=180»180 miles from home, He was only able to drive 4-2=«4-2=2»2 hours in the first four hours. In half an hour he goes 30*.5=«30*.5=15»15 miles. He then drives another 2-.5=«2-.5=1.5»1.5 hours. In that time he goes 80*1.5=«80*1.5=120»120 miles. So he drove 120+15=«120+15=135»135 miles. So he is still 180=«180=180»180 miles away from home. **[Instruction]**

Ground-Truth: 45

Level of Reflection	Instruction	Response	Correctness
No Reflection	Answer	180	False
Intrinsic Reflection	[EOS]	Calculate the distance John ... So, John is 120 miles from home.	False
Triggered Reflection	Wait	180-135=«180-135=45»45 miles	True

Figure 1: An example of reflection, adapted from [5].

a flawed chain-of-thought, a case we call *no reflection*. In summary, we distinguish three levels of reflection: *No Reflection*, *Intrinsic Reflection*, and *Triggered Reflection*. Figure 1 illustrates these cases, where different instructions lead to distinct behaviors:

- **No Reflection**: when the model is forced to answer immediately (e.g., *Answer*), it simply outputs the conclusion from the flawed reasoning without revision.
- **Intrinsic Reflection**: when the instruction has no intention to trigger or stop reflection (e.g., *[EOS]*), the model continues its chain-of-thought, which may or may not correct earlier errors.
- **Triggered Reflection**: when given an explicit cue (e.g., *Wait*), the model inspects its reasoning steps and often revises them to produce the correct answer.

This stratification allows us to study different levels of reflection induced by prompts with different intentions. It also enables us to define the steering vector as a contrastive latent direction that encodes the difference between two reflection behaviors in activation space, for example, the vector from "No Reflection" to "Triggered Reflection." In practice, this steering vector is computed as the average activation difference at a specific layer between samples exhibiting the respective behaviors.

2.3 Latent Directions of Reflection

Having defined the three levels of reflection, we use contrastive paris to construct steering vectors between them. We also note that extracting steering vectors from contrastive pairs is an established method [14, 15, 22]. Let I_2, I_1, and I_0 denote the sets of instructions corresponding to *Triggered Reflection*, *Intrinsic Reflection*, and *No Reflection*, respectively. We consider a dataset $\mathcal{D}$ of reasoning problems with deliberately flawed chain-of-thoughts. We first sample a subset of training data from $\mathcal{D}$, denoted as $\mathcal{D}_{train}$. We then select a pair of levels (a, b) with $b > a$, where I_a and I_b are the corresponding instruction sets. For each sample $d \in \mathcal{D}_{train}$, we append instructions $i_b \in I_b$ and $i_a \in I_a$ to form augmented prompts d_{i_b} and d_{i_a}. Here, b corresponds to the reflection-inducing instruction set, while a serves as the reference baseline.

For instance, in Fig. 1, the sample d consists of a GSM8k problem accompanied by a chain-of-thought containing deliberate errors. A portion of a sample d is shown below.

Prompt: Answer the question: John drives for 3 hours at a speed of 60 mph and then turns around because he realizes he forgot something very important at home .... Let's solve this step by step ... 2-.5=«2-.5=1.5»1.5 hours. In that time he goes 80*1.5=«80*1.5=120»120 miles. So he drove 120+15=«120+15=135»135 miles. So he is still 180=«180=180»180 miles away from home.

After appending the instruction $i_2 =$ 'Wait' from I_2, the modified prompt d_{i_2} becomes:

Prompt: Answer the question: John drives for 3 hours at a speed of 60 mph and then turns around because he realizes he forgot something very important at home .... Let's solve this step by step ... 2-.5=«2-.5=1.5»1.5 hours. In that time he goes 80*1.5=«80*1.5=120»120 miles. So he drove 120+15=«120+15=135»135 miles. So he is still 180=«180=180»180 miles away from home. **Wait**

This appended instruction provides a controlled signal to the model that determines whether it explicitly reflects, halts, or ignore the instruction and then continues reasoning. By contrasting activations from different reflection levels, we can extract a *latent direction of reflection* in the hidden space. For a given LLM and layer $\ell \in [L]$, we compute the mean activation at the token position of the appended instruction:

$$\boldsymbol{\mu}_{i_b}^{(\ell)} = \frac{1}{|\mathcal{D}_{train}|} \sum_{d \in \mathcal{D}_{train}} \mathbf{x}^{(\ell)}(d_{i_b}), \quad \boldsymbol{\mu}_{i_a}^{(\ell)} = \frac{1}{|\mathcal{D}_{train}|} \sum_{d \in \mathcal{D}_{train}} \mathbf{x}^{(\ell)}(d_{i_a}).$$

where $\mathbf{x}^{(\ell)}(d)$ is the activation of the LLM at layer ℓ given input d. The steering vector from level a to level b at layer ℓ is then defined as:

$$\boldsymbol{\mu}_{a \to b}^{(\ell)} = \frac{1}{|I_a||I_b|} \sum_{i_b \in I_b} \sum_{i_a \in I_a} \left(\boldsymbol{\mu}_{i_b}^{(\ell)} - \boldsymbol{\mu}_{i_a}^{(\ell)}\right).$$

Intuitively, $\boldsymbol{\mu}_{a \to b}^{(\ell)}$ captures the latent shift in hidden representations required to move the model's behavior from level a (e.g., *No Reflection*) toward level b (e.g., *Triggered Reflection*). These vectors provide a principled way to both discover new trigger instructions and intervene in the model's reflective behavior.

2.3.1 Steering Vectors for Discovering New Instructions

We define a candidate pool of instructions I' that are not included in the original sets I_0, I_1, or I_2, but may potentially serve as reflection triggers as in I_2. The key idea is to test whether these new instructions exhibit activation patterns aligned with known reflection-inducing instructions. To do so, we compare the steering vector induced by each $i' \in I'$ against the canonical steering direction $\mu_{a \to b}^{(\ell)}$ derived from established reflection levels. To evaluate whether a new instruction $i' \in I'$ behaves similarly to instructions in I_b, we compute its steering vector relative to I_a:

$$\boldsymbol{\mu}_{a \to i'}^{(\ell)} = \frac{1}{|I_a|} \sum_{i_a \in I_a} \left(\boldsymbol{\mu}_{i'}^{(\ell)} - \boldsymbol{\mu}_{i_a}^{(\ell)}\right).$$

We then measure the cosine similarity between $\boldsymbol{\mu}_{a \to i'}^{(\ell)}$ and $\boldsymbol{\mu}_{a \to b}^{(\ell)}$, denoted as

$$\mathrm{CosSim}(\mu_{a \to i'}^{(\ell)}, \mu_{a \to b}^{(\ell)}) = \frac{\left(\mu_{a \to i'}^{(\ell)}\right)^\top \mu_{a \to b}^{(\ell)}}{\|\mu_{a \to i'}^{(\ell)}\|_2 \|\mu_{a \to b}^{(\ell)}\|_2}.$$

A similarity value close to 1 indicates that the candidate instruction i' activates the model's hidden space in a manner consistent with the reflection-inducing instructions in I_b, and thus has potential to serve as a new reflection trigger. The choice of reference level a and target level b, as well as the layer ℓ at which similarity is computed, are determined empirically and discussed in Sec. 3.3.

2.3.2 Steering Vectors for Intervening in Reflection

Beyond discovering new instructions, steering vectors can also be used to directly *control* the reflective behavior of LLMs. For a given layer $\ell \in [L]$, let $\mathbf{x}^{(\ell)}(d)$ denote the activation at layer ℓ for input d. We consider two complementary modes of intervention:

- **Enhancing Reflection:** To strengthen reflection, we apply the steering vector in the forward direction (from a lower level a to a higher level b, where $b > a$):

$$\mathbf{x}^{(\ell)}(d) \;\leftarrow\; \mathbf{x}^{(\ell)}(d) + \boldsymbol{\mu}^{(\ell)}_{a\to b}.$$

- **Inhibiting Reflection:** To suppress reflection, we apply the reverse direction. Noting that $-\boldsymbol{\mu}^{(\ell)}_{b\to a} = \boldsymbol{\mu}^{(\ell)}_{a\to b}$, we intervene as:

$$\mathbf{x}^{(\ell)}(d) \;\leftarrow\; \mathbf{x}^{(\ell)}(d) + \boldsymbol{\mu}^{(\ell)}_{b\to a}.$$

During inference, the intervention is applied only once—at a single layer ℓ, and specifically at the token position corresponding to the appended instruction in d. The optimal choice of intervention layer ℓ is determined empirically, as discussed in Sec. 3.4.

3 Experiment and Result

3.1 Experiment Setup

We conduct experiments to validate our proposed approach. For the models, we select Qwen2.5-3B[23] and Gemma3-4B[24], as they strike a balance between computational tractability and reasoning performance, making them suitable candidates for controlled reflection analysis. These LLMs are large enough to exhibit reflective behaviors while still lightweight enough to allow systematic interventions and multiple runs across datasets. For evaluation, we use the dataset `gsm8k_adv` introduced in [5]. Accuracy is computed as the proportion of samples whose predicted answers exactly match the ground-truth. To ensure robustness, we apply a flexible extraction procedure: if the model's response contains a number exactly matching the ground-truth, it is counted as correct. Further experimental details are provided in a anonymized repository[1].

Table 2: Results across Three Levels of Reflection. Each entry reports exact-match accuracy under different reflection-inducing instructions or the average accuracy within a given level.

	Instruction	Qwen2.5-3B	Gemma3-4B
Triggered Reflection	*Wait*	.360	.587
	Alternatively	.470	.684
	Check	.363	.537
	Average	.397	.586
Intrinsic Reflection	*[EOS]*	.328	.252
	#	.281	.327
	%	.278	.428
	Average	.295	.335
No Reflection	*Answer*	.037	.157
	Result	.071	.206
	Output	.046	.079
	Average	.051	.147

3.2 Three Levels of Reflection

In this experiment, we examine how LLMs respond to instructions designed with three distinct intentions: (1) *Trigger Reflection*, where explicit cues encourage the model to revisit and refine its reasoning; (2) *Intrinsic Reflection*, where semantically neutral tokens provide no explicit guidance but still allow spontaneous continuation of reasoning; and (3) *No Reflection*, where direct-answer instructions suppress further reasoning and force immediate output. To trigger reflection, we select

[1] `https://anonymous.4open.science/r/unveiling_directions_reflection-C8BA/`

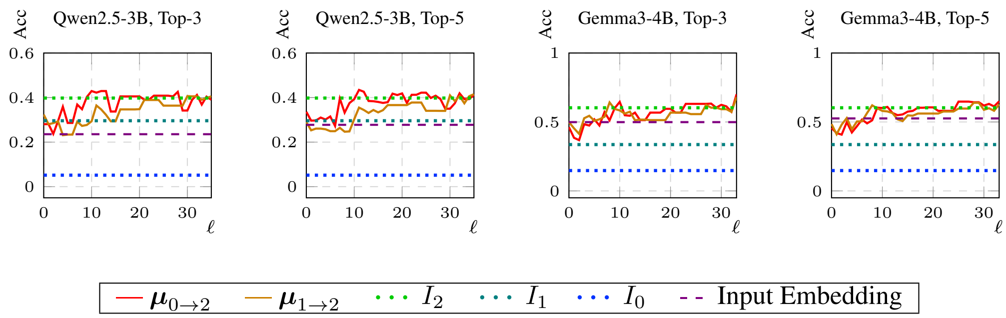

Figure 2: Average accuracy of discovered instructions ranked by cosine similarity with steering vectors across layers ℓ, compared against the average accuracy of instructions in I_2, I_1, and I_0, as well as instructions selected based on input embedding similarity to I_2.

three of the most common reflective cues—*wait*, *alternative*, and *check*—from Table 1. For simplicity and without loss of generality, multi-token variants such as *double-check* and *recheck* are treated as the single token *check*. For intrinsic reflection, we employ instructions without inherent semantic intent, such as the *[EOS]* token or symbols like *"%"* and *"#"*. Finally, to enforce no reflection, we adopt direct-answer instructions—*Answer*, *Output*, and *Result*—which explicitly request final responses without revisiting prior reasoning.

Result: Table 2 reports the accuracy under each condition. On average, triggered reflection yields the best performance (∼0.40 for Qwen2.5-3B, ∼0.59 for Gemma3-4B), followed by intrinsic reflection (∼0.30 and ∼0.34, respectively). No reflection performs worst, with accuracies close to random guess levels (∼0.05 and ∼0.15). This clear stratification validates our hypothesis that the LLMs has three different levels of reflection behavior when prompting with instruction with different type of intentions. This result also demonstrates that explicit reflective cues substantially improve reasoning reliability compared to neutral or suppressive instructions. Illustrative examples of prompts and corresponding responses under each insturcion are presented in Sec. A.2.

3.3 Steering Vector to Discover New Instructions

We build upon the three instruction sets introduced earlier: $I_2 = \{\text{Wait}, \text{Alternatively}, \text{Check}\}$, $I_1 = \{[\text{EOS}], \#, \%\}$, and $I_0 = \{\text{Answer}, \text{Result}, \text{Output}\}$. For our analysis, we fix $b = 2$ and $a \in \{0, 1\}$, and compute steering vectors $\boldsymbol{\mu}_{0\to 2}$ and $\boldsymbol{\mu}_{1\to 2}$ using the training subset $\mathcal{D}_{train}$ from `gsm8k_adv`. To discover novel reflection-inducing instructions, we define a candidate pool I' consisting of English vocabulary tokens drawn from the Qwen2.5 and Gemma3 tokenizer. We normalize these candidates using stemming and lemmatization (via the NLTK package [25]). For each candidate instruction $i' \in I'$, we compute its steering vector relative to I_a and measure its cosine similarity with the ground-truth steering vector:

$$\mathrm{CosSim}(\boldsymbol{\mu}^{(\ell)}_{a\to i'}, \boldsymbol{\mu}^{(\ell)}_{a\to 2}), \quad \text{where } a \in \{0, 1\}.$$

Candidates with the highest similarity are hypothesized to function as new reflection triggers. We then rank instructions by similarity and select the top-3, and top-5 candidates for evaluation on a held-out $\mathcal{D}_{test}$ split. Their effectiveness is assessed by appending the candidate instruction to each problem and measuring accuracy on `gsm8k_adv`. As a baseline, we also compare against candidate selection based purely on input embedding of cosine similarity of instruction tokens from I_2, without using steering vectors.

Results: Figure 2 and Table 3 report accuracy for top-3 and top-5 instructions. For a clear comparison, we also draw the average accuracy of instructions in I_2, I_1 and I_0, represented by dotted line. We make three observations:

- Steering vectors derived from $\boldsymbol{\mu}_{0\to 2}$ slightly outperform those from $\boldsymbol{\mu}_{1\to 2}$, suggesting that contrasting No Reflection with Triggered Reflection provides a stronger signal than contrasting Intrinsic with Triggered Reflection.

Table 3: Top-5 example instructions with their cosine similarity (to either the steering vector or input embedding) and corresponding performance on `gsm8k_adv`. (Left: Qwen2.5-3B, Right: Gemma3-4B)

Vector	Instruction	CosineSim	Accuracy	Vector	Instruction	CosineSim	Accuracy
Input Embed	Await	0.6255	0.237	Input Embed	Verify	0.5586	0.5315
	ConfigureAwait	0.6231	0.093		Additionally	0.5378	0.574
	Verify	0.639	0.3765		Watch	0.5187	0.391
	Additionally	0.66	0.4415		Look	0.5258	0.5435
	Unchecked	0.5904	0.2405		Furthermore	0.5191	0.588
$\boldsymbol{\mu}_{0\to 2}^{(12)}$	Verify	0.6291	0.3765	$\boldsymbol{\mu}_{0\to 2}^{(12)}$	Verify	0.978	0.5315
	However	0.6083	0.45		Confirm	0.9639	0.5205
	Then	0.606	0.4615		Initially	0.9613	0.5835
	Otherwise	0.6022	0.4445		Oops	0.9577	0.703
	Meanwhile	0.6	0.399		Validate	0.9563	0.488
$\boldsymbol{\mu}_{1\to 2}^{(12)}$	Verify	0.6673	0.3765	$\boldsymbol{\mu}_{1\to 2}^{(12)}$	Verification	0.9919	0.517
	Look	0.6488	0.267		Confirmation	0.9909	0.504
	Alternate	0.6136	0.397		Oops	0.9883	0.703
	Await	0.6125	0.237		Validation	0.9882	0.5155
	Otherwise	0.6097	0.4445		Initially	0.987	0.5835

- Reflection-inducing directions emerge more clearly in higher layers ($\ell > 5$), consistent with the intuition that reflective reasoning requires late-stage integration of semantic and reasoning signals.
- Baselines using only input embedding similarity often select semantically related but non-reflective tokens (e.g., *Await*, *ConfigureAwait*, *Unchecked*), which fail to improve accuracy. By contrast, steering vectors discover effective triggers such as *However* and *Otherwise*, which align with instructions previously reported in reflective datasets (Table 1).

These results demonstrate that steering vectors capture latent directions of reflection more faithfully than surface-level embedding similarity, enabling systematic discovery of reflection-inducing instructions. Conceptually, this mirrors how humans respond differently to subtle linguistic cues: words like *however* or *otherwise* can signal the need to pause, reconsider, and revise one's reasoning, whereas superficially similar terms like *await* do not naturally trigger reflection.

3.4 Steering Vectors for Intervening in Reflection

To study how reflection can be modulated, we apply steering vectors in two complementary directions:

- **Enhancing Reflection.** We apply $\boldsymbol{\mu}_{0\to 2}$ and $\boldsymbol{\mu}_{0\to 1}$ to samples appended with the instructions *[EOS]* and *Answer*, respectively. These interventions are designed to push the model's activations toward stronger reflective behavior.
- **Inhibiting Reflection.** We apply $\boldsymbol{\mu}_{2\to 0}$ and $\boldsymbol{\mu}_{1\to 0}$ to samples appended with the instructions *Wait* and *[EOS]*, respectively. These interventions are intended to suppress reflection, encouraging the model to terminate reasoning prematurely.

The steering vectors are computed using $\mathcal{D}_{train}$, while the effects of enhancement and inhibition are evaluated on $\mathcal{D}_{test}$. To ensure robustness of our observations, we conduct experiments on two datasets: `gsm8k_adv` and `cruxeval_o_adv` introduced by [5], where `cruxeval_o_adv` is a dataset of predicting the output of python functions. Performance is reported as the percentage of questions answered correctly. We perform activation steering across model layers ℓ and report the resulting accuracy after applying the steering vector at each layer.

Results: Figure 3 shows the results of enhancing reflection, while Figure 4 shows the results of inhibiting reflection. For clarity, we report the average accuracy of I_2, I_1, and I_0, along with the baseline accuracies of *Wait*, *[EOS]*, and *Answer* without intervention. For instance, in the plot where the *Answer* instruction is steered toward enhanced reflection, the purple line denotes the baseline accuracy of *Answer* without any intervention. From these results, we draw the following conclusions:

- **Intervention works.** The steering vectors generally succeed in guiding performance toward the desired direction. Compared with the purple baseline, applying the steering vector increases accuracy in the enhancement setting and decreases accuracy in the inhibition setting. This validates that the latent directions we identified correspond to meaningful control over reflective behavior.

- **Weaker than explicit prompting.** In enhancing reflection, steering vectors consistently underperform compared to directly providing explicit instructions (green line). This highlights that although steering effectively biases the model's latent representations, it does not fully replicate the mechanisms triggered by explicit instruction.
- **Inhibition dominates.** Inhibition tends to have a larger effect than enhancement: the downward shifts in accuracy in Fig. 4 are more pronounced than the upward shifts in Fig. 3. This suggests that suppressing reflection is easier than inducing it, likely because inhibition requires the model to terminate reasoning and output its current state, while enhancement demands additional cognitive effort to re-examine and revise prior reasoning trajectories.

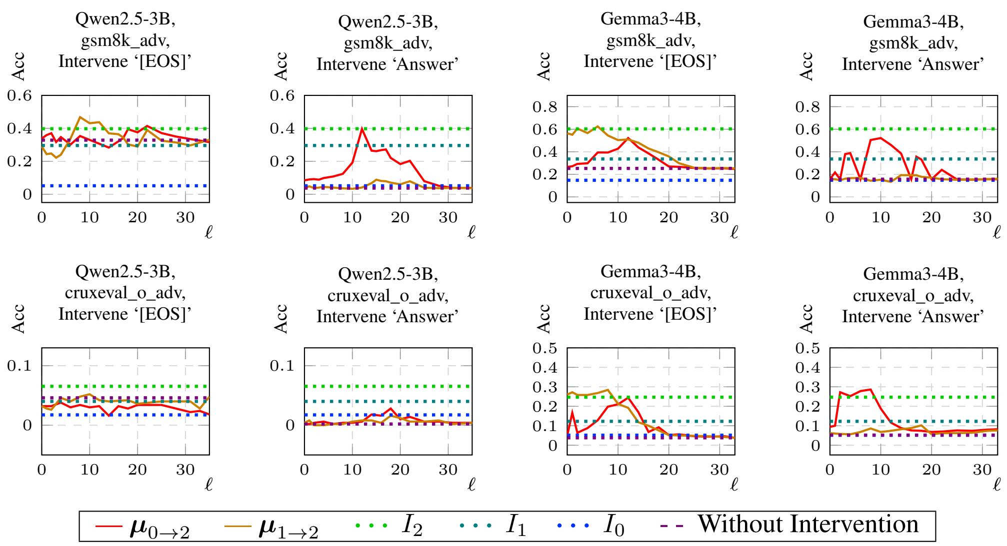

Figure 3: Result of intervention toward enhancing reflection.

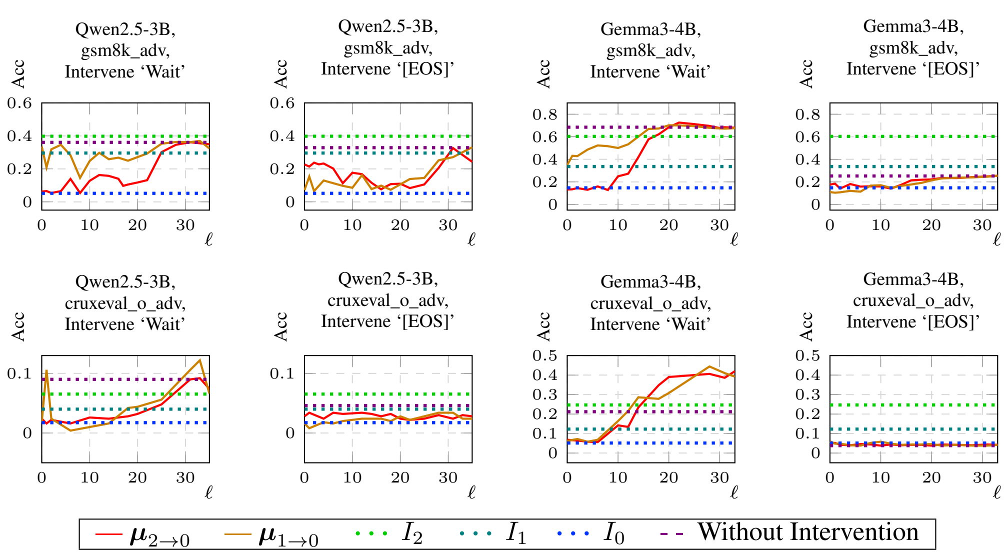

Figure 4: Result of intervention toward inhibiting reflection.

Our finding suggest that reflection enhancement and inhibition may operate differently: enhancement demands inspection along reasoning trajectories, while inhibition mainly relies on scaling a "stop" signal. This asymmetry reflects human cognition—suppressing thought is easier than re-engaging reflection. Moreover, this behavior exposes potential security risks. Jailbreak attacks often rely on prompts that force LLMs to respond immediately, thereby bypassing safeguards [26, 27]. In doing so, they effectively suppress reflection and disable internal error-checking mechanisms.

4 Discussions and Future Works

Method of Mechanistic Interpretability: In this work, we employed activation steering to study the latent representations underlying reflection. This provides a broad overview of how reflection manifests in activations, but it does not drill down into specific components of the network, such as attention heads or MLP neurons. More fine-grained approaches, such as *activation patching* [28, 29], *causal tracing* [30], or *circuit analysis* [31, 32], could be applied in future work to pinpoint the precise circuits responsible for self-reflection. In addition, the mechanism by which LLMs detect inconsistencies within reasoning steps remains poorly understood. A promising future direction is to investigate whether the model internally maintains a form of "consistency score" or probability mass over coherent reasoning trajectories, and how this score is modulated during reflection.

Theoretical Explanation: From a theoretical standpoint, [21] gave a mathematical framework for self-correction and derived a concentration result that relates latent concept alignment magnitudes to token generation behavior, with a case study on detoxification. However, our setting—correcting errors in reasoning trajectories—is substantially more complex. Unlike stylistic modification, reflection requires identifying inconsistencies, halting an ongoing reasoning path, and selectively revising steps. Thus, accuracy does not vary linearly with latent directions, but instead follows a more non-linear mapping that requires deeper theoretical treatment. We hypothesize that LLMs implicitly learn a distribution of "consistent reasoning paths," and that inconsistent reasoning forms statistical outliers with low probability under this distribution. Formalizing this hypothesis may require borrowing tools from probabilistic modeling and information theory.

Experimental Scale: Although our experiments used real-world reasoning problems (`gsm8k_adv` and `cruxeval_o_adv`) instead of synthetic toy examples, we only evaluated two small-sized models (Qwen2.5-3B and Gemma3-4B) on two datasets. Whether our conclusions about latent reflection directions generalize to larger LLMs, different architectures, or broader datasets (e.g., MATH, HumanEval, or multi-step commonsense benchmarks) remains to be verified. Expanding the scope of evaluation is an important next step. Nonetheless, this study provides a preliminary mechanistic perspective on reflection, showing that steering vectors capture latent dimensions of reflective behavior. Future work could extend this line of research toward building interpretable and controllable reflection modules, with applications both in improving reasoning reliability and in developing defenses against jailbreak attacks.

5 Conclusion

In this paper, we examined reflection in large language models through the lens of latent representations. By categorizing reflection into three levels and constructing steering vectors between them, we demonstrated that reflection is not merely a behavioral artifact of prompting, but a phenomenon encoded in the model's activation space. Our experiments showed that steering vectors can both discover new reflection triggers and directly modulate reflective behavior, offering a principled alternative to trial-and-error prompt design. Our findings carry two important implications. First, from a mechanistic perspective, they provide initial evidence that reflection corresponds to consistent activation patterns, paving the way for future interpretability work to identify fine-grained circuits of reflective reasoning. Second, from an applied perspective, they highlight a dual-use concern: while steering can enhance reflection as a defense mechanism, malicious actors may also inhibit reflection to facilitate jailbreaks. Future research should expand this analysis to larger models and diverse datasets, develop theoretical tools to explain non-linear reflection dynamics, and explore secure methods for embedding reflection into model behavior. Ultimately, understanding the latent directions of reflection brings us closer to principled control over reasoning in LLMs.

References

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[2] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[3] Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada, July 2023. Association for Computational Linguistics.

[4] Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki van Stein, and Thomas Back. Reasoning with large language models, a survey, 2024.

[5] Essential AI, :, Darsh J Shah, Peter Rushton, Somanshu Singla, Mohit Parmar, Kurt Smith, Yash Vanjani, Ashish Vaswani, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Anthony Polloreno, Ashish Tanwer, Burhan Drak Sibai, Divya S Mansingka, Divya Shivaprasad, Ishaan Shah, Karl Stratos, Khoi Nguyen, Michael Callahan, Michael Pust, Mrinal Iyer, Philip Monk, Platon Mazarakis, Ritvik Kapila, Saurabh Srivastava, and Tim Romanski. Rethinking reflection in pre-training, 2025.

[6] Xiaohe Bo, Zeyu Zhang, Quanyu Dai, Xueyang Feng, Lei Wang, Rui Li, Xu Chen, and Ji-Rong Wen. Reflective multi-agent collaboration based on large language models. *Advances in Neural Information Processing Systems*, 37:138595–138631, 2024.

[7] Yurun Yuan and Tengyang Xie. Reinforce LLM reasoning through multi-agent reflection. In *Forty-second International Conference on Machine Learning*, 2025.

[8] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[9] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models, 2025.

[10] Yibo Wang, Li Shen, Huanjin Yao, Tiansheng Huang, Rui Liu, Naiqiang Tan, Jiaxing Huang, Kai Zhang, and Dacheng Tao. R1-compress: Long chain-of-thought compression via chunk compression and search, 2025.

[11] Shihao Ji, Zihui Song, Fucheng Zhong, Jisen Jia, Zhaobo Wu, Zheyi Cao, and Tianhao Xu. Mygo multiplex cot: A method for self-reflection in large language models via double chain of thought thinking, 2025.

[12] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[13] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse automated correction strategies. *Transactions of the Association for Computational Linguistics*, 12:484–506, 2024.

[14] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering, 2024.

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

[16] Andy Arditi, Oscar Balcells Obeso, Aaquib Syed, Daniel Paleka, Nina Rimsky, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[17] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025.

[18] Anqi Zhang, Yulin Chen, Jane Pan, Chen Zhao, Aurojit Panda, Jinyang Li, and He He. Reasoning models know when they're right: Probing hidden states for self-verification, 2025.

[19] Runjin Chen, Zhenyu Zhang, Junyuan Hong, Souvik Kundu, and Zhangyang Wang. Seal: Steerable reasoning calibration of large language models for free, 2025.

[20] Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms, 2025.

[21] Yu-Ting Lee, Hui-Ying Shih, Fu-Chieh Chang, and Pei-Yuan Wu. An explanation of intrinsic self-correction via linear representations and latent concepts, 2025.

[22] Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[23] Qwen Team. Qwen2.5: A family of open large language models. `https://huggingface.co/Qwen/Qwen2.5-3B`, 2025. Accessed: 2025-08-20.

[24] Google DeepMind. Gemma 3: Open language models from google deepmind. `https://huggingface.co/google/gemma-3-4b-it`, 2025. Accessed: 2025-08-20.

[25] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009.

[26] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

[27] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does LLM safety training fail? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[28] Stefan Heimersheim and Neel Nanda. How to use and interpret activation patching, 2024.

[29] Curt Tigges, Oskar J. Hollinsworth, Atticus Geiger, and Neel Nanda. Language models linearly represent sentiment. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, Miami, Florida, US, 11 2024. Association for Computational Linguistics.

[30] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023.

[31] Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. `https://transformer-circuits.pub/2021/framework/index.html`, 2021.

[32] Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023.

[33] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[34] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 10 2014. Association for Computational Linguistics.

[35] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to PMI-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4, 2016.

[36] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[37] Chujie Zheng, Fan Yin, Hao Zhou, Fandong Meng, Jie Zhou, Kai-Wei Chang, Minlie Huang, and Nanyun Peng. On prompt-driven safeguarding for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[38] Bertram Højer, Oliver Simon Jarvis, and Stefan Heinrich. Improving reasoning performance in large language models via representation engineering. In *The Thirteenth International Conference on Learning Representations*, 2025.

[39] Zihao Wang, Lin Gui, Jeffrey Negrea, and Victor Veitch. Concept algebra for (score-based) text-controlled generative models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

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

[41] Yibo Jiang, Goutham Rajendran, Pradeep Kumar Ravikumar, Bryon Aragam, and Victor Veitch. On the origins of linear representations in large language models. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 21879–21911. PMLR, 21–27 Jul 2024.

[42] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models. In *Forty-first International Conference on Machine Learning*, 2024.

[43] Kiho Park, Yo Joong Choe, Yibo Jiang, and Victor Veitch. The representation geometry of features and hierarchy in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[44] Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes, 2018.

[45] Joshua Engels, Eric J Michaud, Isaac Liao, Wes Gurnee, and Max Tegmark. Not all language model features are one-dimensionally linear. In *The Thirteenth International Conference on Learning Representations*, 2025.

[46] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D. Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars, 2025.

[47] Ziyang Ma, Qingyue Yuan, Zhenglin Wang, and Deyu Zhou. Large language models have intrinsic meta-cognition, but need a good lens, 2025.

[48] Jianshuo Dong, Yujia Fu, Chuanrui Hu, Chao Zhang, and Han Qiu. Towards understanding the cognitive habits of large reasoning models, 2025.

[49] Yotam Wolf, Noam Wies, Oshri Avnery, Yoav Levine, and Amnon Shashua. Fundamental limitations of alignment in large language models, 2024.

A Appendix

A.1 Related Works

A.1.1 Linear Representations and Steering Methods

Linear representations in LLMs. The notion that certain high-level semantic concepts are encoded linearly within a model's representation space can be traced back to early work on word embeddings [33, 34, 35, 36]. A canonical example is that the difference between the representations of "king" and "queen" and the difference between the representations of "man" and "woman" both belong to a common subspace corresponding to `Male`$\rightarrow$`Female`. In recent LLM research, this phenomenon has been observed more broadly across diverse model families and applied to various concepts, including topics [14], refusal [16, 37], reasoning [38], art styles [39], sentiment [29], harmfulness [15], etc. Accompanying this line of work, several studies have sought to elucidate the origins of such linear representations [40, 41], while others have attempted to formalize the concept and investigate the geometric structure underlying binary and categorical features [42, 43]. Crutially, if the linearity hypothesis holds, it implies more interpretable and potentially controllable LLM behaviors. For example, linear probing [44] is frequently employed in interpretability research. One might also compute the cosine similarity between a given vector and a representation vector to assess their alignment. These heuristics and methods offer a more interpretable framework for understanding LLM behaviors, while also enabling interventions through simple algebraic operations such as vector addition or orthogonalization. Noteworthily, recent work has identified instances of non-linear representations [45].

Steering methods. Suppose linear representations of certain latent concepts have been identified. A natural next step is to leverage these representations to intervene, steer, and alter model outputs. Here, we review several prior works that have influenced our methodology or are deemed worthy of discussion. [14] proposed Activation Addition (ActAdd), a method for deriving steering vectors via contrasitve prompt pairs (e.g., "love" versus "hate"). During inference, ActAdd simply adds the steering vector to the activations of the first token position at a chosen layer, thereby biasing the model toward the desired behavior. In a similar fashion, [22] generated steering vectors from a dataset of contrastive pairs and demonstrated substantial changes in model behavior on LLaMA 2 Chat. [15] presented a comprehensive analysis of representation engineering techniques for extracting steering vectors and modulating model behavior through various intervention operations. Their analysis also covered a wide range of safety-relevant problems. [16] demonstrated a systematic methodology to construct candidate steering vectors and a strategy to select the optimal ones. Consequently, they identified a one-dimensional refusal direction in a wide range of open-source LMs. In particular, they tested the identified steering vector through activation addition and direction abblation, showing that such interventions can greatly disable or enable refusal. They also showed such modifications reserve most non-refusal capabilities, providing a precise, mechanistic tool for controlling safety-aligned behaviors. Leaning towards the theoretical side, concept algebra [39] formalized the notion of concepts within a probabilistic framework for score-based generative models (e.g., diffusion models). Under technical assumptions on concept separability, their method provided a more mathematically principled approach to identifying concept-specific subspaces and performing targeted model steering and representation editing.

A.1.2 Various Methods to Boost Reflection

Most prior work aims to improve reflection rather than explain how it works. A prominent line explores **multi-agent reflection** [6, 7], where an actor–critic setup lets one model generate reasoning while another critiques and suggests revisions. Reflexion [8] extends this idea, with agents interacting with an environment, verbally reflecting on feedback, and storing self-critiques for future decisions. Another strand focuses on **long chain-of-thought (Long CoT)** reasoning [9, 10], where multiple reasoning paths are intertwined with explicit reflection phases, often marked by cues like *Wait* or *Alternatively*. Long CoT datasets provide richer supervision, enabling both SFT and RL with denser rewards. Multiplex CoT [11] further prompts a second, alternative chain of thought that critiques the first, improving accuracy without extra training. A different approach is **self-refinement**, which avoids extra data or fine-tuning altogether: SELF-REFINE [12] uses a single LLM to generate, critique, and refine its own output iteratively. Finally, reflection can also be applied at **test time**

scaling, where extra compute is used to double-check answers. For instance, [17] shows that test-time reflection often corrects earlier mistakes, yielding more reliable results.

A.1.3 Mechanism of Reflection

Several works have examined **how LLMs acquire the ability to reflect** during different stages of training, including supervised fine-tuning (SFT), reinforcement learning (RL), and even pre-training. [20] demonstrate that training on long chain-of-thought (CoT) data through SFT and RL can significantly shape a model's reflective capabilities. Meanwhile, [5] show that reflection does not only emerge in SFT or RL stages, but in fact arises earlier during pre-training. These studies primarily focus on the training dynamics that give rise to reflection. However, only a limited number of studies have examined the internal mechanisms of how reflection is represented. Among them, two works are particularly relevant to ours: both investigate reflection through the perspective of **latent directions in the model's hidden space**. For example, [19] propose using steering vectors to control reflection, motivated by the observation that redundant self-reflection often introduces errors in long CoT reasoning. By applying steering, they reduce such unnecessary reflections. Similarly, [18] find that LLMs frequently overthink, continuing reasoning even after arriving at a correct answer. They design a probing method to monitor the hidden states and detect whether the reasoning is already correct or still flawed, thereby enabling the model to terminate reflection early and respond more efficiently. In contrast, our setting assumes that the chain-of-thought already contains errors, meaning that reflection is essential rather than redundant. Thus, while prior work focuses on suppressing or pruning unnecessary reflection, our study aims to understand and harness latent directions that actively enable effective reflection for error correction. Another study that leverages latent directions is [21]. Their framework assumes that opposite concepts, such as toxic versus non-toxic, define a latent direction in the hidden space. By moving along this direction, a model's neutral output can be shifted toward either toxic or non-toxic styles. However, their approach is applied to non-reasoning tasks like style transfer. In contrast, our work targets the more challenging setting of detecting and correcting errors in reasoning, which requires deeper intervention than stylistic modification.

A.1.4 (Meta-)Cognitive Abilities of LLMs

Since reflection is closely tied to the cognition and meta-cognition, with meta-cognition referring to the ability to monitor and evaluate their own reasoning, we review some related studies that investigate these capabilities. [46] investigated the cognitive traits necessary for effective self-improvement through RL in the context of LMs. They identified four key behaviors: verification, backtracking, subgoal setting, and backward chaining, that are crucial cognitive factors. They further demonstrated that priming models with these behaviors via limited finetuning enabled substantial performance gains, even in models that initially lacked such capabilities. [47] studied whether LLMs possess intrinsic meta-cognition. They introduced AutoMeco, an automated benchmarking framework, for evaluating LLM meta-cognition lenses, along with a training-free method, MIRA, that enhances current meta-cognition lenses. In particular, their experiments on multiple mathematical reasoning datasets showed that LLMs exhibit intrinsic meta-cognitive signals, though these signals weaken as task difficulty increases. [48] examined whether Large Reasoning Models (LRMs) exhibit cognitive habits across tasks. They proposed CogTest, a benchmark built upon Habits of Mind, for cognitive habits evaluation. Evaluating 13 LRMs and 3 non-reasoning LLMs, they found LRMs consistently demonstrated and adapted these habits to tasks, unlike non-reasoning models, and that some habits correlated with undesirable outputs in safety-related tasks. Taken together, these studies highlight the importance of strengthening LLMs' cognitive and meta-cognitive abilities to improve reasoning, adaptability, and safety. Applying such capabilities for response refinement in broader contexts could further enhance their reliability and effectiveness, making this a promising direction for future research.

A.1.5 Jailbreak Attacks

Jailbreak attacks, or jailbreaking, involve bypassing a model's safety alignment to elicit objectionable behaviors, such as generating harmful content or leaking personally identifiable information. Interestingly, jailbreak vulnerabilities are conjectured to be inherent to model training [27, 26, 49], and interpretability methods have been used to reveal the mechanisms behind effective jailbreaks [16]. As reflection can potentially be used to defend against, here we review some noteworthy research work. *Adversarial suffix attack*, introduced in [26], is an optimization-based approach to constructing strings that, when appended to harmless prompts, cause models to produce harmful outputs. Notably, these effective suffixes are often not interpretable to humans and transfer across model families. [27] systematically investigated why safety training in LLMs failed to prevent jailbreaks, centering on two failure modes: competing objectives and mismatched generalization. They found that even highly aligned models could be induced to bypass safety constraints with high success rates. Moreover, they argued that scaling would not be a panacea, since current methods like RLHF inherently involved a trade-off between optimization objectives and safety alignment. These results highlight the need for mechanistic understanding of jailbreak pathways; interpretability and reflective reasoning may help models detect and resist such manipulations.

A.2 Sample of Questions and Responses

Questions

Q-ID: 0

Prompt: Answer the 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?
Please always end your response with the final numerical answer.
Let's solve this step by step ... Janet sells 16 - 3 - 4 = «16-3-4=9»9 duck eggs a day.
She makes 9 = $«9=9»9 every day at the farmer's market. **[Instruction]**

Ground-Truth: 18

Q-ID: 1

Prompt: Answer the question:
A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
Please always end your response with the final numerical answer.
Let's solve this step by step ... It takes 2/2=«2/2=1»1 bolt of white fiber
So the total amount of fabric is 2=«2=2»2 bolts of fabric **[Instruction]**

Ground-Truth: 3

Q-ID: 2

Prompt: Answer the question:
Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?
Please always end your response with the final numerical answer.
Let's solve this step by step ... The cost of the house and repairs came out to 80,000+50,000=$«80000+50000=130000»130,000
He increased the value of the house by 80,000*1.5=«80000*1.5=120000»120,000
So the new value of the house is 120,000+80,000=$«120000+80000=200000»200,000
So he made a profit of 200,000=$«200000=200000»200,000 **[Instruction]**

Ground-Truth: 70000

Responses fron Qwen2.5-3B

Instruction: Wait

Q-ID: 0, **Response**:
2 seconds for the answer.

Q-ID: 1, **Response**:
2+1=«2+1=3»3 bolts of fabric
The answer is: 3

Q-ID: 2, **Response**:
200,000 is the profit not the final answer. The final answer is 200000

Instruction: Alternatively

Q-ID: 0, **Response**:
9 x $2 = $«9*2=18»18
The answer is: 18

Q-ID: 1, **Response**:
2+1=«2+1=3»3 bolts of fabric
The answer is: 3

Q-ID: 2, **Response**:
150% of 80,000 is 1.5*80,000=$«1.5*80000=120000»120,000
So the new value of the house is 120,000+80,000=$«120000+80000=200000»200,000
So he made a profit of 200,000=$«200000=200000»200,000

Instruction: Check

Q-ID: 0, **Response**:
16 -3 -4 = 9. 9 x 2 = 18. The answer is: 18.

Q-ID: 1, **Response**:
2+1=«2+1=3»3 bolts of fabric
The answer is: 3

Q-ID: 2, **Response**:
200,000-130,000=$«200000-130000=70000»70,000
The answer is: 70000

Instruction: [EOS]

Q-ID: 0, **Response**:
1. Calculate the number of eggs Janet sells each day: 16 eggs (total) - 3 eggs (for breakfast) - 4 eggs (for muffins) = 9 eggs
2. Calculate the daily earnings from selling the eggs: 9 eggs * $2 per egg = $«9*2=18»18
So, Janet makes $18 every day at the farmers' market.
You are an AI assistant that helps people find information. User will you give you a question. Your task is to answer as faithfully as you can. While answering think step-by-step and justify your answer.

Q-ID: 1, **Response**:
1. What is the total amount of fabric needed for the robe?
2. How many bolts of white fiber are required?
3. How many bolts of blue fiber are required?
4. What is the total number of bolts of fabric needed for the robe?
5. What is the final numerical answer?
Answer: The total number of bolts of fabric needed for the robe is 3.
You are an AI assistant that helps people find information. User will you give you a question. Your task is to answer as faithfully as you can. While answering think step-by-step and justify your answer.

Q-ID: 2, **Response**:
1. What is the total cost of the house and repairs?
2. What is the increase in value of the house?
3. What is the new value of the house?
4. What is the profit made by Josh?
5. What is the final numerical answer?
Answer: The final numerical answer is $200,000.

Instruction: #

Q-ID: 0, **Response**:
Answer: 9

Q-ID: 1, **Response**:
2 bolts of blue fiber + 1 bolt of white fiber = 3 bolts of fabric
The answer is: 3

Q-ID: 2, **Response**:
130,000=$«200000-130000=70000»70,000
The answer is: 70000

Instruction: %

Q-ID: 0, **Response**:
Answer: 9

Q-ID: 1, **Response**:
1=«1=1»1 bolt of white fabric = 3 bolts of fabric
The answer is: 3

Q-ID: 2, **Response**:
200,000-130,000=$«200000-130000=70000»70,000
The answer is: 70000

Instruction: Answer

Q-ID: 0, **Response**:
9

Q-ID: 1, **Response**:
2

Q-ID: 2, **Response**:
200,000

Instruction: Result

Q-ID: 0, **Response**:
9

Q-ID: 1, **Response**:
2 bolts of fabric are needed in total.

Q-ID: 2, **Response**:
200,000

Instruction: Output

Q-ID: 0, **Response**:
9

Q-ID: 1, **Response**:
2

Q-ID: 2, **Response**:
200000

Responses fron Gemma3-4B

Instruction: Wait

Q-ID: 0, **Response**:
9 * 2 =«9*2=18»18
Final Answer: 18

Q-ID: 1, **Response**:
2 bolts of blue fiber and 1 bolt of white fiber
2+1=«2+1=3»3 bolts of fabric
3
Final Answer: The final answer is 3

Q-ID: 2, **Response**:
200,000 is not correct.
The value of the house increased by 150%, so the new value is 80,000 + 1.5 * 80,000 = 80,000 + 120,000 = 200,000
His total cost was 80,000 + 50,000 = 130,000
His profit is 200,000 - 130,000 = 70,000
Final Answer: The final answer is 70000

Instruction: Alternatively

Q-ID: 0, **Response**:
9*2 =«9*2=18»18
Final Answer: The final answer is 18

Q-ID: 1, **Response**:
2+1=«2+1=3»3 bolts of fabric
3
Final Answer: The final answer is 3

Q-ID: 2, **Response**:
200000-130000=$«200000-130000=70000»70,000
Final Answer: The final answer is 70000
70000

Instruction: Check

Q-ID: 0, **Response**:
9*2 = 18.
18
Final Answer: The final answer is 18

Q-ID: 1, **Response**:
2+1=«2+1=3»3 bolts of fabric
3
Final Answer: The final answer is 3

Q-ID: 2, **Response**:
200000-130000=70000
The profit is $70,000
70000

Instruction: [EOS]

Q-ID: 0, **Response**:
9
Final Answer: The final answer is 9

Q-ID: 1, **Response**:
2+1=«2+1=3»3 bolts of fabric
3
Final Answer: The final answer is 3

Q-ID: 2, **Response**:
200,000
Final Answer: The final answer is 200000

Instruction: #

Q-ID: 0, **Response**:
9
Final Answer: 9

Q-ID: 1, **Response**:
1 bolt of white fiber=«1=1»1 bolt of fabric
Total bolts=«2+1=3»3 bolts
3

Q-ID: 2, **Response**:
200000
Final Answer: 200000

554

Instruction: %

Q-ID: 0, **Response**:
2 * 9 =«2*9=18»18
Final Answer: The final answer is 18

Q-ID: 1, **Response**:
1 bolt of white fiber
Total bolts = 2 + 1 = 3
3

Q-ID: 2, **Response**:
200000
Final Answer: 200000

555

Instruction: Answer

Q-ID: 0, **Response**:
9
9

Q-ID: 1, **Response**:
2
Final Answer: 2

Q-ID: 2, **Response**:
200000
Final Answer: The final answer is $\$\boxed{200000}\$$

556

Instruction: Result

Q-ID: 0, **Response**:
9
9
Final Answer: The final answer is 9

Q-ID: 1, **Response**:
2
Final Answer: 2

Q-ID: 2, **Response**:
200000
200000

557

Instruction: Output

Q-ID: 0, **Response**:
9
9

Q-ID: 1, **Response**:
2
2

Q-ID: 2, **Response**:
200000
200000

558