

Interpreting and Controlling LLM Reasoning through Integrated Policy Gradient

Changming Li¹ Kaixing Zhang¹ Haoyun Xu¹ Yingdong Shi¹ Zheng Zhang¹ Kaitao Song² Kan Ren^{1*}

Abstract

Large language models (LLMs) demonstrate strong reasoning abilities in solving complex real-world problems. Yet, the internal mechanisms driving these complex reasoning behaviors remain opaque. Existing interpretability approaches targeting reasoning either identify components (e.g., neurons) correlated with special textual patterns, or rely on human-annotated contrastive pairs to derive control vectors. Consequently, current methods struggle to precisely localize complex reasoning mechanisms or capture sequential influence from model internal workings to the reasoning outputs. In this paper, built on *outcome-oriented* and *sequential-influence-aware* principles, we focus on identifying components that have sequential contribution to reasoning behavior where outcomes are cumulated by long-range effects. We propose Integrated Policy Gradient (IPG), a novel framework that attributes reasoning behaviors to model’s inner components by propagating compound outcome-based signals such as post reasoning accuracy backward through model inference trajectories. Empirical evaluations demonstrate that our approach achieves more precise localization and enables reliable modulation of reasoning behaviors (e.g., reasoning capability, reasoning strength) across diverse reasoning models.

1. Introduction

Large Language Models (LLMs) have shown outstanding performance in addressing natural language processing (NLP) tasks (Touvron et al., 2023; Brown et al., 2020; Qwen Team, 2024). Beyond simple next-token prediction, modern LLMs now demonstrate sophisticated reasoning abilities, including structured, step-by-step problem solving (Wei et al., 2022; Khot et al., 2023). Despite these advancements, the internal reasoning mechanisms underlying large language

¹ShanghaiTech University ²Independent Researcher. Correspondence to: Kan Ren <renkan@shanghaitech.edu.cn>.

Preprint. February 3, 2026.

models, especially how they represent and encode such capability, remain unclear. This lack of transparency poses challenges for truthfulness, safety, and controllability in reasoning systems (Wang et al., 2025b).

Mechanistic interpretability studies suggest that the internal representation space of LLMs encodes reasoning-related concepts. Existing interpretability approaches for reasoning fall into two categories. The first analyzes individual internal components, such as neurons (Rai & Yao, 2024) or Sparse Autoencoder (SAE) features (Galichin et al., 2025; Fang et al., 2026), applying mediation analysis (Stolfo et al., 2023) on *short-term effect*, e.g., arithmetic, or on correlations between their activations and text patterns, e.g., reasoning-related tokens. The second derives control vectors in the representation space to induce desired behaviors (Højer et al., 2025; Wang et al., 2025a), but these methods rely heavily on well-designed contrastive *input* pairs, making it difficult to guarantee that the intended target states are genuinely elicited (Højer et al., 2025).

These prevalent approaches face one or both of the following limitations: (i) *Reliance on co-occurrences*: they rely solely on input prompts or activation co-occurrences with text-pattern, failing to isolate the components that drive reasoning behaviors. (ii) *Neglecting sequential influence*: they capture only short-term effects, e.g., arithmetic, neglecting the cumulative sequential influence necessary for multi-step reasoning (Sui et al., 2025), such as task correctness.

To address the limitations, we design our approach for mechanistic interpretability of reasoning in LLMs based on two key principles: *outcome-oriented* and *sequential-influence-aware*. The effects of internal components should be evaluated by their contribution to reasoning behaviors, and these contributions should be measured with respect to the outcome of reasoning. Since reasoning unfolds over long horizons, the outcome (e.g., correctness of the answer) captures the integrated influence of intermediate steps. Both principles further enable effective steering of LLM reasoning.

Building on these principles, we propose Integrated Policy Gradient (IPG), a novel training-free framework for interpreting and controlling reasoning in LLMs. IPG first applies gradient-based attribution to identify influential internal *components* (such as neurons or SAE features) within

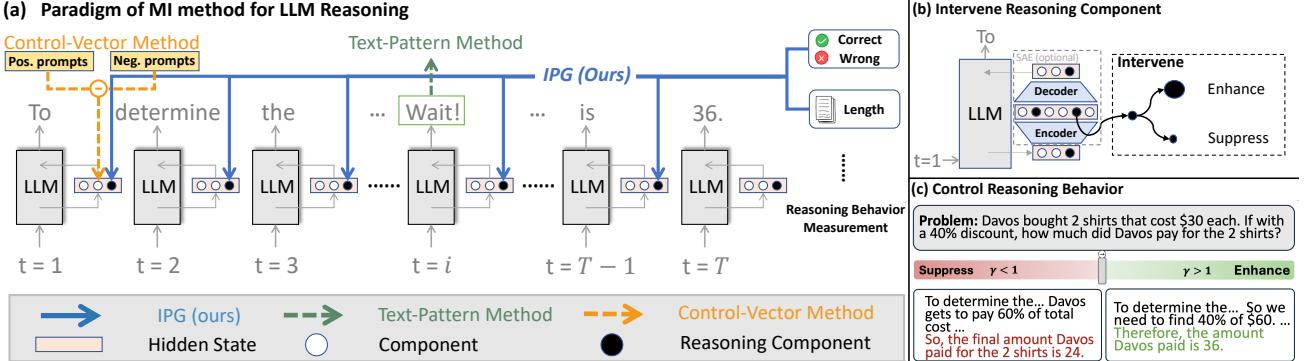


Figure 1. Paradigms of interpretability method for reasoning in LLMs and comparison results on reasoning datasets. (a) Illustration of the three paradigms, regarding (i) text-pattern methods that associate high activations with special tokens (e.g., “Wait”); (ii) control-vector methods that rely on human-craft contrastive input pairs; and (iii) our proposed Integrated Policy Gradient method. t is the reasoning time step. (b) Intervene reasoning component: selected components are scaled by a factor γ (see Section 3.3). (c) Control reasoning behavior with both suppressing and enhancing.

the LLM that shape reasoning behavior. However, standard gradient approaches are limited because reasoning outcomes are typically non-differentiable, e.g., final answer correctness provided by external verifier (Guo et al., 2025). To address this, IPG incorporates policy gradient (Mnih et al., 2016; Schulman et al., 2017), which back-propagates outcome-based behavioral signals through the entire inference trajectory to the model’s internal components. IPG further accumulates policy gradients across a linearly interpolated path between a baseline (e.g., zero value) to the target activation for each component, improving completeness (Sundararajan et al., 2017) in attributing reasoning behaviors. Importantly, IPG achieves this efficiently *without updating the model parameters*. Figure 1 illustrates the differentiation of the current paradigms and our methods.

We evaluate the components identified by IPG on representative open-source LLMs and demonstrate our superior *control effectiveness* in reasoning behaviors, including reasoning capability and reasoning strength (Mondorf & Plank, 2024; Sheng et al., 2025). Moreover, these components exhibit strong *transferability*, results derived from prompt-elicited reasoning models (Yang et al., 2024) can be directly applied to training-induced variants (Guo et al., 2025) within the same model family, eliminating redundant interpretation efforts. Finally, individual components enable *fine-grained interpretability* over distinct aspects of reasoning, including problem understanding and task decomposition, underscoring the potential of IPG for advancing mechanistic interpretability of LLM reasoning.

In summary, our main contributions are threefold, centered on improving the mechanistic understanding and controllability of reasoning in LLMs. (i) We introduce Integrated Policy Gradient (IPG), a novel training-free framework towards reasoning mechanism that is both *sequential-influence-aware* and *outcome-oriented*, attributing reasoning behaviors directly to internal components based on long-

horizon outcomes. (ii) Our approach extends to various internal units, such as individual neurons and SAE features (Gao et al., 2025), providing a unified tool to interpret and modulate the reasoning behavior of LLMs. (iii) We demonstrate that IPG achieves superior control effectiveness, enabling reliable steering of various reasoning behaviors and fine-grained interpretability over specific reasoning aspects.

2. Related work

Reasoning in Large Language Models. Reasoning has become a core capability of Large Language Models (LLMs). Existing improvements in LLM’s reasoning ability include mainly prompt-elicited (Wei et al., 2022; Khot et al., 2023) and training-induced reasoning (Guo et al., 2025; OpenAI, 2024; Qwen Team, 2024). Training-induced methods involve reinforcement learning (RL) to optimize reasoning abilities with outcome reward signals. Alternatively, other methods utilize supervised fine-tuning, where models are explicitly trained on datasets of reasoning chains (Yue et al., 2023) or distilled from more powerful reasoning models (Guo et al., 2025). Despite notable improvements in reasoning capabilities, the internal representations and encoding of reasoning in these models remain unclear.

Mechanistic Interpretability in Language Models for Reasoning. Mechanistic interpretability (MI) seeks to reverse-engineer the computations of language models by examining their internal representations (Bereska & Gavves, 2024). Reasoning, however, requires multi-step and long-horizon internal computation, posing unique challenges for the prevalent MI methods (Plaat et al., 2025). Existing work targets reasoning-related concepts for internal components interpretation, including arithmetic (Rai & Yao, 2024) neurons and reason-specific features (Galichin et al., 2025; Fang et al., 2026) using sparse autoencoders (Gao et al., 2025; Huben et al., 2024), focusing on correlating activations with surface text patterns (e.g., reasoning tokens like “wait”).

Other work aims to control model behavior by adding vectors derived from contrastive input pairs (Højer et al., 2025; He et al., 2026; Sheng et al., 2025; Ma et al., 2026). These methods require carefully annotated contrastive pairs and prompt design, which struggle to ensure the intended target states are truly elicited and makes it challenging to reliably align the reasoning behaviors (Højer et al., 2025). In addition, intervention-based methods like Causal Mediation Analysis (CMA) (Stolfo et al., 2023) can identify causal modules in reasoning, but they primarily focus on short-term tasks such as simple arithmetic. Overall, existing methods fail to establish a clear sequential relationship between target reasoning behaviors and internal components, and struggle to capture the cumulative, multi-step influences inherent in the reasoning process. In contrast, our approach is *sequential-influence-aware* and *outcome-oriented*, capable of attributing reasoning behavior without relying on human annotations or hand-crafted text patterns.

3. Methodology

3.1. Preliminary: Reasoning in Large Language Models

3.1.1. PROBLEM SETUP

Reasoning Task Formulation. Let F be a pretrained large language model. Formally, we define LLM reasoning as a sequential generation process. Given an input query \mathbf{x} , the language model first generates a trajectory $\tau = (a_1, a_2, \dots, a_t, \dots, a_T)$ of length T autoregressively, consisting of generated tokens a_t , and produces a final answer y . We denote the overall reasoning process as $(\tau, y) = F(\mathbf{x})$. This formulation characterizes reasoning as a *multi-step* decision-making process with long-term sequential dependency, where each reasoning step can be associated with various reasoning behaviors.

Reasoning Component Space. The internal representation space of LLMs encodes concepts related to reasoning behaviors (Galichin et al., 2025). Our framework supports localization of these concepts on either neuron-level hidden state component z_i , or feature-level component f_i by integrating with Sparse Autoencoders (SAEs) (Makhzani & Frey, 2014) to obtain a disentangled space. Let $\mathbf{z}_t^l = [z_{t,1}^l, \dots, z_{t,n}^l] \in \mathbb{R}^n$ denote the hidden state at a specific layer l given an input \mathbf{x} . We obtain the representation of the feature space $\mathbf{f}^l = [f_{t,1}^l, \dots, f_{t,m}^l] \in \mathbb{R}^m$ by encoding \mathbf{z}_t^l through an SAE. Details are provided in Section B.1. Together, these representations constitute our *reasoning component space*. We define a unified variable $\mathbf{h}_t^l \in \mathbb{R}^d$ to represent the state under analysis, where $\mathbf{h}_t^l = [h_{t,1}^l, \dots, h_{t,d}^l] \in \{\mathbf{z}_t^l, \mathbf{f}_t^l\}$. Consequently, $d = n$ when analyzing the neuron-level hidden state, and $d = m$ is for the feature-level space.

Building on this formulation, we decompose the reasoning process into *trajectory-level behaviors* and their underlying

component-level mechanisms. Concretely, at each generation step t along the trajectory τ , the model instantiates an internal representation \mathbf{h}_t^l . This representation mediates the generation dynamics, governing both the immediate next-token prediction and the accumulation of long-horizon dependencies. Formally, we express the generation of the reasoning trajectory τ and the final response y as a function of the input \mathbf{x} conditioned on the sequence of internal states. For notational simplicity, we omit layer l to use \mathbf{h}_t to represent target component in subsequent part.

$$(\tau, y) = F(\mathbf{x} \mid \mathbf{h}_{1:T}^l). \quad (1)$$

Crucially, the formulation reveals that reasoning outcomes emerge from the cumulative contribution of internal components across the full trajectory, a property we term *sequential-influence*. It considers how a component affects the multi-step decisions that lead to the final outcome (τ, y) .

3.1.2. MEASUREMENT OF REASONING BEHAVIOR

Reasoning behavior is the model’s underlying mechanism during the reasoning process (i.e., how reasoning unfolds along the multi-step trajectory), including reasoning capability (Mondorf & Plank, 2024) and reasoning strength (Sheng et al., 2025). Based on outcome-oriented principle, we define a signal function $J(\cdot)$ to further help assess the model’s inner workings regarding various reasoning behaviors. Specifically, J can be instantiated as a binary reward $J(\tau) \in \{0, 1\}$ on outcome to measure reasoning capability, or as the total number of generated tokens $J(\tau) = |\tau|$ to quantify reasoning strength, where τ denotes a reasoning trajectory. Notably, the definition of $J(\cdot)$ is highly *flexible*, as it can be readily extended to capture more specific reasoning behaviors, such as reflection, as well as domains other than reasoning, including safety. Further discussion is provided in Section D.8.

3.2. Integrated Policy Gradient (IPG) Method

Prior interpretability methods often rely on text-pattern or contrastive pairs. However, these strategies face at least one of the limitations in reasoning contexts: (i) failure to capture cumulative, multi-step dependencies (Højer et al., 2025); (ii) lack of strict *sequential-influence* links between internal activations and long-horizon outcomes (Rai & Yao, 2024); (iii) standard gradient-based attribution struggle with sparse, non-differentiable feedback (Dai et al., 2022) (e.g., final-answer correctness).

To address these challenges, we now introduce the Integrated Policy Gradient (IPG) method, a framework proceeds by extending policy gradients to the representation space, refining the attribution via path integration, and ensuring robustness through dataset-level aggregation. The process is depicted in Figure 1 (a).

Policy Gradients in Representation Space. We first integrate policy-gradient methods (Mnih et al., 2016; Schulman et al., 2017), which propagate outcome-aware signals (e.g., correctness rewards) backward through the trajectory, thereby enabling attribution over long-horizon reasoning effects. To obtain reliable attribution of reasoning behavior, we extend the perspective of policy gradient (Mnih et al., 2016; Schulman et al., 2017) from parameter space to representation space by propagating outcome-based signals back to internal components \mathbf{h}_t . Concretely, for a signal function $J(\cdot)$ that assesses reasoning behavior, with \mathbf{h} as the variable of interest, the policy gradient in representation space is

$$\frac{\partial J}{\partial \mathbf{h}} = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^T \frac{\partial}{\partial \mathbf{h}_t} \log \pi_\theta(a_t | s_t; \mathbf{h}_t) \cdot A^\pi(s_t, a_t) \right], \quad (2)$$

where s_t is the prefix tokens and a_t is the next token at the reasoning time step t . $\tau = (a_1, \dots, a_T)$ is a reasoning trajectory following π_θ which is the policy parameterized by θ in the LLM F that maps from s_t to a_t . $A^\pi(s_t, a_t)$ is the advantage function estimating the long-horizon benefit of taking a_t under s_t . Empirically, this value is estimated based on the feedback derived from the signal function $J(\cdot)$.

Crucially, rather than using policy gradients to update model parameters θ , we compute gradients with respect to \mathbf{h} to obtain attribution signals that reflect the long-horizon impact of internal activations on reasoning outcomes. These signals form the basis for quantifying their effects on reasoning behavior and selecting components for subsequent causal interventions (Section 3.3).

Path Integral Attribution. Raw gradients can be noisy that capture only local sensitivity at a specific point. To address this, we propose Integrated Policy Gradient (IPG), guided by our *outcome-oriented, sequential-influence-aware* principle. The integral accounts for the global contribution of a component by aggregating its influence across the entire interpolation path from a baseline to its original value.

Specifically, to quantify the influence of each component $h_{t,i}$ at step t on a reasoning-behavior measure J , we define the attribution score of reasoning behavior as

$$\text{IPG}(i; \mathbf{x}) = \sum_{t=1}^T (h_{t,i} - h'_{t,i}) \int_0^1 \mathbb{E}_{\tau \sim \pi_\theta} \left[\frac{\partial}{\partial h_{t,i}} \log \pi_\theta(a_t | s_t; \mathbf{h}'_t + \alpha(\mathbf{h}_t - \mathbf{h}'_t)) \cdot A^\pi(s_t, a_t) \right] d\alpha. \quad (3)$$

where \mathbf{h}'_t is the relative baseline value of \mathbf{h}_t at step t . Deriving from Equation (2), for each generated trajectory $\tau = (a_1, \dots, a_T)$, $\log \pi_\theta(a_t | s_t; \mathbf{h}'_t + \alpha(\mathbf{h}_t - \mathbf{h}'_t))$ is the log-probability of selecting token a_t given the context s_t under the interpolated hidden state, which is weighted by

the advantage $A^\pi(s_t, a_t)$. We accumulate policy gradients along the path from a baseline \mathbf{h}' to the original activation \mathbf{h} , spanning across activation levels (Sundararajan et al., 2017; Dhamdhere et al., 2019). This path-integral construction yields baseline-aware importance scores that reduce gradient noise and ensure that the total attribution accounts for the entire change in policy output, producing robust signals for causal intervention. We provide ablation in Section D.3.

To ensure robustness, the IPG attribution score is computed per sample and aggregated across a dataset. Let $\mathcal{D} = \{\mathbf{x}^{(d)}\}_{d=1}^M$ be a small supporting dataset of M reasoning questions. For each component index i and each sample index d , we compute the per-sample attribution score $\text{IPG}(i; \mathbf{x}^{(d)})$ (Equation (3)). These scores are aggregated into a global importance statistic, for example, the mean attribution score S_i for component h_i , and the top- p components $\mathbf{P} = \{i_1, \dots, i_p\}$ are treated as the most influential components

$$S_i = \frac{1}{M} \sum_{d=1}^M \text{IPG}(i; \mathbf{x}^{(d)}), \quad \mathbf{P} = \arg \max_{i \in [n]} S_i. \quad (4)$$

Discussion: Efficiency and Generality of IPG. Importantly, IPG method is *computation-efficient* and *data-efficient*. The attribution step is performed *once* for a given frozen model. Once the set \mathbf{P} is identified, interventions can be applied repeatedly without recomputing attributions. In addition, we conclude that even with a relative small dataset \mathcal{D} , our IPG can accurately locate the component that influentially drive reasoning capability (Section D.1.2). Besides, \mathbf{h} can be any interpretable module within the model. Our IPG can be seamlessly integrated with various policy gradient algorithms such as GRPO (Shao et al., 2024) to help better estimate reasoning outcomes (Section A.3).

3.3. Control Reasoning Behavior via IPG

To causally control the reasoning behavior inside the model, we intervene on the components indexed by $i \in \mathbf{P}$ identified in Section 3.2, following interpretability practices (Dai et al., 2022). Formally, for each component $h_{t,i}$ at step t , we apply a multiplicative scaling factor as

$$h_{t,i} = \text{Intervene}(h_{t,i}) = \gamma h_{t,i} \quad (5)$$

By setting different values of γ , we can enhance ($\gamma > 1$) or suppress ($0 \leq \gamma < 1$) the identified components, thereby controlling the reasoning behavior in the target model (in Figure 1 (b)).

Finally, at each step t , we substitute the original internal state \mathbf{h}_t with the modified state vector $\tilde{\mathbf{h}}_t$ that is composed of the intervened components \tilde{h}_i and original components $h_{j \notin \mathbf{P}}$. This yields a modified generation process, producing

Table 1. Comparison of IPG and baseline methods on enhancing and suppressing accuracy across reasoning datasets for Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) and Llama3.1-8B-Instruct (Grattafiori et al., 2024). Best and second best results are shown in **bold** and underlined format. The arrows \uparrow and \downarrow means the higher or lower results are better, respectively.

Method	Enhancing Accuracy \uparrow				Suppressing Accuracy \downarrow			
	GSM8K	MATH-500	AIME-2024	GPQA Diamond	GSM8K	MATH-500	AIME-2024	GPQA Diamond
Qwen2.5-Math-1.5B Instruct								
Original	82.41	63.00	10.00	24.75	82.41	63.00	10.00	24.75
Random	82.86	63.20	<u>16.67</u>	26.77	79.15	59.80	6.67	27.78
Activation	82.56	63.40	20.00	25.75	67.78	39.80	0.00	<u>15.15</u>
R.N. (Rai & Yao, 2024)	82.26	62.20	13.33	24.24	<u>74.75</u>	58.00	0.00	<u>15.15</u>
C.V. (Højer et al., 2025)	83.85	62.60	<u>16.67</u>	28.78	-	-	-	-
SAE-R (Galichin et al., 2025)	83.32	64.00	13.33	25.76	82.26	63.80	10.00	21.71
IPG (Ours)	<u>84.61</u>	<u>65.80</u>	20.00	<u>33.33</u>	10.69	25.00	0.00	12.12
IPG-SAE (Ours)	84.76	66.00	20.00	35.35	81.73	<u>28.80</u>	<u>3.00</u>	20.20
Llama3.1-8B-Instruct								
Original	85.89	41.80	6.67	23.23	85.89	41.80	6.67	23.23
Random	86.20	41.40	6.67	27.78	84.98	39.00	<u>10.00</u>	26.77
Activation	85.75	<u>43.00</u>	<u>13.33</u>	25.25	81.88	28.60	0.00	14.14
R.N. (Rai & Yao, 2024)	86.05	40.80	6.67	21.21	75.44	<u>2.06</u>	0.00	9.60
C.V. (Højer et al., 2025)	85.97	40.00	6.67	<u>28.78</u>	-	-	-	-
SAE-R (Galichin et al., 2025)	86.28	41.80	10.00	26.77	85.97	43.20	<u>10.00</u>	22.22
IPG (Ours)	<u>86.81</u>	42.40	<u>13.33</u>	27.27	50.42	1.40	0.00	0.00
IPG-SAE (Ours)	87.41	44.00	20.00	29.80	<u>71.19</u>	33.40	0.00	<u>1.52</u>

a new trajectory $\tilde{\tau}$ and response \tilde{y} :

$$(\tilde{\tau}, \tilde{y}) = F(\mathbf{x} | \tilde{\mathbf{h}}_{1:T}). \quad (6)$$

Figure 1 (c) visualizes this effect, and we empirically verify in Sections 4.2 and 4.4 that the modified trajectory reflects the targeted modulation of reasoning behavior.

4. Experiment

In this section, we conduct empirical evaluation for our IPG in interpreting and controlling reasoning behaviors in LLMs. We aim to answer the below research questions (**RQs**). **RQ1**: Can IPG identify internal components that exhibit verifiable *sequential-influence* effects on reasoning behaviors? **RQ2**: Does IPG find the general and consistent reasoning components across benchmarks? **RQ3**: Can IPG find components that reflect more granular reasoning ability? The reproducible code will be released soon.

4.1. Experimental Setup

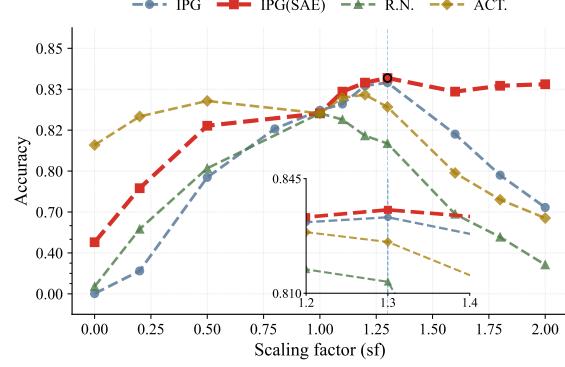
Evaluation Benchmarks. For comprehensive evaluation, we choose several reasoning-related benchmarks, including GSM8K (Cobbe et al., 2021), MATH500 (Hendrycks et al., 2021), AIME2024 (Mathematical Association of America, 2024) and GPQA-Diamond (Rein et al., 2023).

Target Models. We experiment with two representative open-source LLMs: Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) and Llama3.1-8B-Instruct (Grattafiori et al., 2024), and additionally evaluate DeepSeek-Qwen-1.5B (Guo et al., 2025). This selection covers different model scales, architectures (Qwen vs. LLaMA), and reasoning paradigms (prompt-elicited vs. training-distilled),

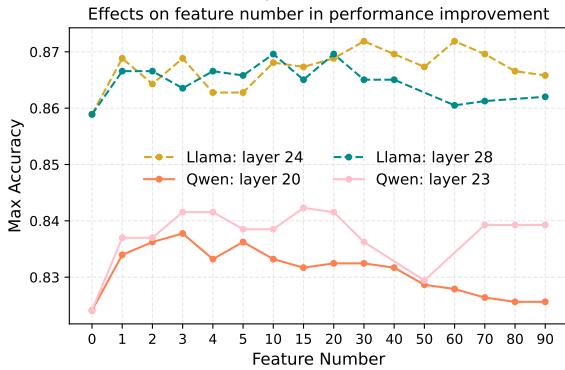
highlighting the generalizability of our method.

Baselines. We compare our IPG and IPG-SAE (integrated with k-SAE (Gao et al., 2025)) against several baselines: *Random* (RND.), which samples neurons uniformly, and *Activation* (ACT.), which ranks by activation magnitude. We also include *Reasoning Neuron* (R.N.) (Rai & Yao, 2024), an SAE-based method (SAE-R) (Galichin et al., 2025) and *Control Vector* (C.V.) (Højer et al., 2025), which represent interpretable approaches using text patterns and contrastive data pairs for reasoning-related interventions. More details are in Appendix C.2.

Implementation Details. We constrain our IPG and baseline methods on the residual stream following (Højer et al., 2025; Galichin et al., 2025) for fair comparison. For the policy gradient algorithm, we employ GRPO (Shao et al., 2024). The external verifier for reasoning outcomes comes from either rule-function based (Appendix A.3), such as generation length (Appendix D.2) or model-based signals for accuracy (using Skywork-Reward-V2-Llama-3.1-8B (Liu et al., 2025)). We integrate k-SAE (Gao et al., 2025) into target models, configured with an expansion factor of 16 and $k = 32$ active features. Following prior work (Cheng et al., 2024; Tang et al., 2025), we consistently employ a greedy decoding strategy during inference with zero-shot setting. For the baseline h'_i mentioned in Equation (3), we set $h'_i = 0$ for all $i \in [n]$, where n is the hidden dimension, following (Dai et al., 2022). We also provide analysis of choice of baseline in Section D.1.4. For intervention, we apply a positive scaling factor γ for enhancement and set $\gamma = 0$ for suppression, as described in Section 3.3. More details are provided in Appendix B.



(a) Scaling factor effects.



(b) Feature number effects.

Figure 2. Effects of targeted interventions on reasoning performance. Left: impact of the intervention scaling factor on Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024). Right: impact of the number of selected features on reasoning accuracy.

4.2. Identification and Control of Reasoning Behaviors

In this section, we evaluate our IPG and baseline methods both qualitatively and quantitatively on the sequential-influence effectiveness of reasoning behavior control across different reasoning-related datasets (**RQ1**).

4.2.1. QUANTITATIVE RESULTS ON CONTROLLING REASONING BEHAVIOR

Analysis 1: Reasoning behavior control with identified components. We evaluate our IPG alongside baseline methods for controlling reasoning behavior in two types of language models across multiple reasoning datasets in Table 1. As Control Vector (Højer et al., 2025) is not applicable to the suppression setting, its results are excluded.

Findings 1: IPG identify internal components with sequential-influence effects on reasoning behaviors. In Table 1, IPG achieves excellent and consistent control over the performance of reasoning tasks in both enhancement and suppression settings. Integrating with SAEs further increases our effectiveness. R.N. (Rai & Yao, 2024) and SAE-R (Galichin et al., 2025) provide only modest improvements, while C.V. (Højer et al., 2025) shows mixed gains and fails on MATH500 (Hendrycks et al., 2021). These

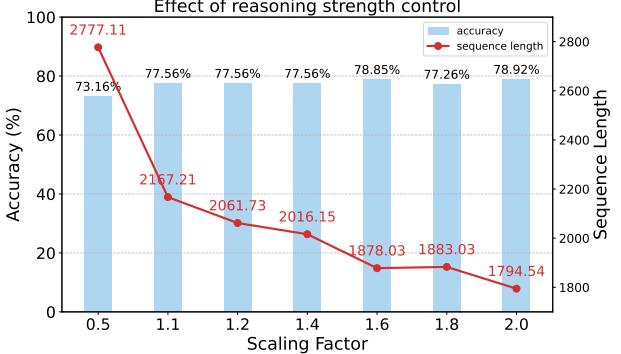


Figure 3. Reasoning strength control in GSM8K (Cobbe et al., 2021) for DeepSeek-R1-Distilled-Qwen-1.5B (Guo et al., 2025). IPG identifies neurons that is related with reasoning length, suggested by effective control while maintaining high accuracy.

results suggest the limitations of approaches that rely on text-patterns or input contrastive pairs. While our IPG offers a more robust framework for identifying reasoning components with influential effects and steering reasoning behavior than baseline methods. We observe that suppressing neurons identified by IPG causes precipitous performance declines, indicative of the polysemantic nature of individual neurons (Mu & Andreas, 2020; Olah et al., 2020).

Analysis 2: Fine-grained control of reasoning behaviors. We mainly control two reasoning behaviors: reasoning capability and reasoning strength (Sheng et al., 2025). For reasoning capability, we evaluate the sensitivity of model performance by varying the scaling factor γ and the number of intervened components (top- p SAE features), as shown in Figure 2. Specifically, we analyze the top-ranked features based on our attribution score (Equation (4)) across two randomly selected layers in each model (Figure 2b). For reasoning strength, we vary the scaling factor γ to examine how it modulates the reasoning chain length.

Finding 2: IPG enables precise modulation of various reasoning behaviors. As shown in Figure 2a, baselines yield only minor or unstable effects, whereas IPG produces clear and predictable controllability that amplifying the identified components consistently enhances reasoning performance, while suppressing them causes degradation. Notably, IPG with SAE remains stable even as baselines collapse. In Figure 2b, we can observe that reasoning-critical features are concentrated in the top-ranked subset, with even single-feature interventions yielding substantial gains. Figure 3 demonstrates that interventions successfully modulate the output length without sacrificing final answer accuracy. This shows that distinct reasoning behaviors are encoded within specific components. Crucially, IPG precisely identifies the components corresponding to these behaviors, enabling interventions that yield predictable influential effects.

Question		
Carlos is planting a lemon tree. The tree will cost \$90 to plant. Each year it will grow 7 lemons, which he can sell for \$1.5 each. It costs \$3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?		
Original	Enhanced Reasoning	Suppressed Reasoning
<p>To determine how many years it will take for Carlos to start earning money...</p> <p>6. **Solving for n:**</p> $n \geq \frac{90}{7.50} = 12 \quad \times$ <p>Therefore, it will take Carlos $\boxed{12}$ years to start earning money on the lemon tree.</p>	<p>To determine how many years it will take for Carlos to start earning money ...</p> <p>6. **Finding the break-even point:** We need to find the smallest n such that the total profit is positive.</p> $7.50 \times n - 90 > 0$ <p>Solving for n: $n > \frac{90}{7.50} = 12 \quad \checkmark$</p> <p>Since n must be a whole number, we round up to the next whole number. Therefore, $n = 13$.</p> <p>Thus, it will take $\boxed{13}$ years before Carlos starts earning money on the lemon tree.</p>	<p>To determine how many years it will take for Carlos to earn money...</p> <p>Setting the total revenue equal to the total costs to find the break-even point:</p> $7.5n = 90 + 3n$ <p>7. **Solve for n:**</p> $\begin{aligned} 4.5n &= 90 \\ n &= \frac{90}{4.5} = 20 \end{aligned} \quad \times \quad \times$ <p>Therefore, it will take $\boxed{20}$ years for Carlos to start earning money from the lemon tree.</p>

Figure 4. Responses generated by Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) on one example in GSM8K (Cobbe et al., 2021), including both original and intervened outputs, with the model’s reasoning ability elicited to arrive at the correct answer.

4.2.2. QUALITATIVE RESULTS ON CONTROLLING REASONING BEHAVIOR

Analysis 3: Response difference before and after intervention. We showcase an example from GSM8K (Cobbe et al., 2021) before and after applying intervention (see in Section 3.3) in the neurons identified by our IPG, demonstrating the impact on enhancing and suppressing reasoning ability, as illustrated in Figure 4. More examples are provided in Section D.6.2.

Finding 3: IPG precisely identifies reasoning-critical components. As shown in Figure 4, the original response fails to follow the strict requirement. The contrast shows that interventions on IPG-identified components have influential effects on intermediate reasoning steps. For the suppressing setting, the model can not even get the equation correct. This demonstrates that IPG does not merely push the model toward a correct final output, it also improves the internal reasoning trajectory, making the reasoning process more complete and coherent. Motivated by this finding, we provide further exploration on the granular aspects of reasoning behaviors, as presented in Section 4.4.

4.3. General Reasoning Mechanism across Tasks

In this section, we evaluate whether the components identified by our IPG and the baseline approach can transfer seamlessly across datasets, with the goal of identifying generalizable and consistent components that genuinely and causally drive the model’s reasoning behavior (RQ2).

Analysis 4: Components consistency across datasets. To investigate how consistent the reasoning components are, we investigate the transferability of these components by applying those derived from GSM8K (Cobbe et al., 2021) to MATH500 (Hendrycks et al., 2021), and vice versa. The resulting accuracy is presented in Table 2. We further quantify

Table 2. Performance on MATH500 (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021), for cross-dataset transfer. Best and second best results are shown in **bold** and underlined format.

Method	MATH-500 → GSM8K		GSM8K → MATH-500	
	Acc. (%) ↑	Avg. Tok.	Acc. (%) ↑	Avg. Tok.
Original	82.41	319.68	63.00	569.35
Activation	82.94	317.17	63.60	567.39
R.N. (Rai & Yao, 2024)	82.87	312.63	63.20	556.97
C.V. (Højer et al., 2025)	83.39	320.03	63.60	568.55
IPG (Ours)	<u>83.78</u>	321.81	65.00	556.49
IPG-SAE (Ours)	84.08	315.86	<u>64.40</u>	585.27

cross-dataset consistency in Figure 8 (Section D.4) using Dice similarity on selected neuron indices P (Section 3.3).

Finding 4: IPG finds consistent reasoning mechanisms across diverse tasks. Cross-dataset transfer results, as shown in Table 2, suggests that IPG achieves the most accuracy gains, showing our consistency across tasks. Comparing with existing interpretability baseline methods, we conclude that our IPG accurately identified the inherent reasoning components, demonstrating stronger cross-benchmark generalization. Additionally, Figure 8 in Section D.4 demonstrates high overlap across the selected component sets. This reflects the nature of the benchmarks that while they diverge considerably in difficulty and elicit distinct reasoning behaviors, they share fundamental underlying skills. Overall, our IPG successfully captures general and transferable reasoning mechanisms while retaining the necessary adaptability specific to the benchmark.

4.4. Fine-grained Reasoning Behavior Discovery

We further analyze the identified components to reveal granular aspects of reasoning behaviors, shedding some light on the mechanisms underlying LLM reasoning (RQ3).

Analysis 5: Granular aspects of reasoning behavior discovery. We analyze reasoning behavior at two finer-grained level, reasoning aspect-level and reasoning process-level. For aspect level, we causally intervene on individual com-

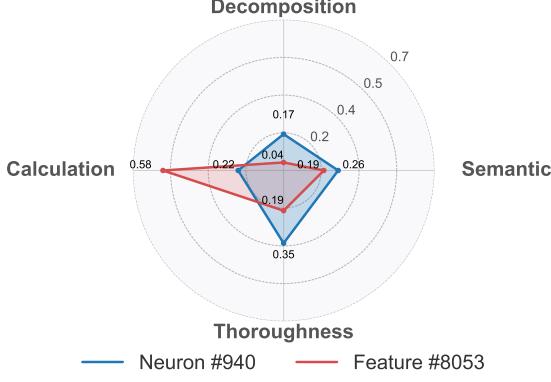


Figure 5. Dual radar plots of neuron- and feature-level interventions: enhancing neuron #940 and feature #8053 on layer 20.

ponents, e.g., single neuron or feature ranked by IPG scores (Equation (4)), and assess their impact on reasoning behavior over a subset of GSM8K (Cobbe et al., 2021). Each component is evaluated along four reasoning dimensions (*Semantic*, *Decomposition*, *Thoroughness*, and *Calculation*) and details are in Table 13 in Section B.3.1. Representative examples of enhancing a single neuron versus an SAE feature are shown in Figure 5. For process-level, we analyze the underlying mechanism of how reasoning unfolds over the course of multi-step reasoning. We examine the temporal activation patterns of representative SAE features across reasoning progress (details in Section D.6.1).

Finding 5: IPG shows granular aspects on reasoning behaviors. As shown in Figure 5, enhancing neuron #940 alters both *Thoroughness* and *Semantic*, whereas boosting SAE feature #8053 primarily improves *Calculation*. Consistent with prior observations of neuron polysemyticity (Mu & Andreas, 2020; Olah et al., 2020), intervening on a single neuron frequently impacts multiple dimensions simultaneously. In contrast, SAE-derived features are more disentangled as they tend to affect a single reasoning aspect, enabling more targeted control. Moreover, feature-level temporal analysis (Figure 6) reveals that such disentanglement also manifests at reasoning process, where different features detected by IPG are activated selectively at different reasoning stages. Additional cases are provided in Section D.6.1.

4.5. Mechanisms transfer under Distilled Models

Analysis 6: Transferring IPG-identified neurons into reasoning-distilled models. Besides general purpose models with prompt-elicted reasoning, we probe the robustness of IPG-identified mechanisms under a reasoning-distilled model. Specifically, we intervene on neurons identified in Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) to DeepSeek-R1-Distilled-Qwen-1.5B (Guo et al., 2025).

Finding 6: Components identified by IPG can be seamlessly transferred. As shown in Table 3, steering the inher-

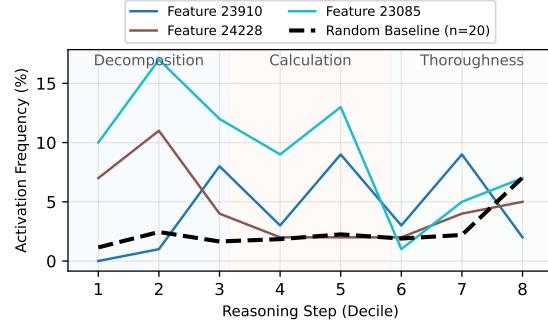


Figure 6. Temporal activation dynamics of representative SAE features (#23085, #23910, #24228) on layer 24, measured across normalized reasoning progress discretized into step deciles.

Table 3. Results of transferred IPG neuron enhancement with reason-distilled models.

Model	GSM8K	MATH500	AIME2024
DeepSeek-R1-Distill-Qwen-1.5B	75.21	61.40	16.67
+ IPG Neuron Enhancement	78.01	62.00	30.00

ited neurons in the DeepSeek-distilled model consistently improves performance across datasets. This indicates that the core reasoning neurons remain crucial even when the underlying model is distilled for reasoning. This result not only validates the ability of IPG to pinpoint fundamental, robust neurons for reasoning, but also offers an insight for the mechanism of model distillation, suggesting that distillation might strengthen the key reasoning-related components that are already present in the base model, rather than reshaping the internal reasoning structure.

5. Conclusion

In this work, we propose IPG, a novel framework for interpreting LLM reasoning behavior that is based on outcome-oriented and sequential-influence-aware principles. Our IPG applies gradient-based methods to identify influential internal components that contribute to reasoning behavior. By incorporating policy gradients (Schulman et al., 2017), we attribute the sparse and non-differentiable outcomes that depend on cumulative and long-range effects. Empirically, IPG provides effective and interpretable control of reasoning behavior across different types of LLMs. Additionally, IPG exhibits transferability both in reasoning datasets and between prompt-elicted and training-induced model variants. Our IPG offers interpretable solution to control reasoning ability in LLMs. Future work includes extending our IPG framework to interpret domains where behaviors lack step-wise, differentiable supervision, such as the emotional intelligence and creativity in LLMs and to precisely manipulate model behavior by steering the identified components.

References

- Belitsky, M., Kopiczko, D. J., Dorkenwald, M., Mirza, M. J., Snoek, C. G. M., and Asano, Y. M. Kv cache steering for inducing reasoning in small language models. *arXiv preprint arXiv:2507.08799*, 2025.
- Bereska, L. and Gavves, S. Mechanistic interpretability for AI safety - a review. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=ePUVetPKu6>. Survey Certification, Expert Certification.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS ’20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- Cheng, X., Li, J., Zhao, W. X., Zhang, H., Zhang, F., Zhang, D., Gai, K., and Wen, J.-R. Small agent can also rock! empowering small language models as hallucination detector, 2024. URL <https://arxiv.org/abs/2406.11277>.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.
- Dai, D., Dong, L., Hao, Y., Sui, Z., Chang, B., and Wei, F. Knowledge neurons in pretrained transformers. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8493–8502, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.581. URL <https://aclanthology.org/2022.acl-long.581/>.
- Dhamdhere, K., Sundararajan, M., and Yan, Q. How important is a neuron. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=SylKoo0cKm>.
- Dutta, S., Singh, J., Chakrabarti, S., and Chakraborty, T. How to think step-by-step: A mechanistic understanding of chain-of-thought reasoning. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=uHLDkQVtC>.
- Fang, Y., Wang, W., Xue, M., Deng, B., Xu, F., Liu, D., and Feng, F. Controllable llm reasoning via sparse autoencoder-based steering. *arXiv preprint arXiv:2601.03595*, 2026.
- Galichin, A., Dontsov, A., Druzhinina, P., Razzhigaev, A., Rogov, O. Y., Tutubalina, E., and Oseledets, I. I have covered all the bases here: Interpreting reasoning features in large language models via sparse autoencoders, 2025. URL <https://arxiv.org/abs/2503.18878>.
- Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL <https://zenodo.org/records/12608602>.
- Gao, L., la Tour, T. D., Tillman, H., Goh, G., Troll, R., Radford, A., Sutskever, I., Leike, J., and Wu, J. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=tcsZt9ZNKD>.
- Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., and et al. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.
- Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

- Han, S., Rao, K., Ettinger, A., Jiang, L., Lin, B. Y., Lambert, N., Choi, Y., and Dziri, N. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in Neural Information Processing Systems*, 37:8093–8131, 2024.
- He, Z., Xiong, G., Liu, B., Sinha, S., and Zhang, A. Reasoning beyond chain-of-thought: A latent computational mode in large language models. *arXiv preprint arXiv:2601.08058*, 2026.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset, 2021. URL <https://arxiv.org/abs/2103.03874>.
- Højer, B., Jarvis, O. S., and Heinrich, S. Improving reasoning performance in large language models via representation engineering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=IssPhpUsKt>.
- Huben, R., Cunningham, H., Smith, L. R., Ewart, A., and Sharkey, L. Sparse autoencoders find highly interpretable features in language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=F76bwRSLeK>.
- Khot, T., Trivedi, H., Finlayson, M., Fu, Y., Richardson, K., Clark, P., and Sabharwal, A. Decomposed prompting: A modular approach for solving complex tasks, 2023. URL <https://arxiv.org/abs/2210.02406>.
- Lindsey, J., Gurnee, W., Ameisen, E., Chen, B., Pearce, A., Turner, N. L., Citro, C., Abrahams, D., Carter, S., Hosmer, B., Marcus, J., Sklar, M., Templeton, A., Bricken, T., McDougall, C., Cunningham, H., Henighan, T., Jermyn, A., Jones, A., Persic, A., Qi, Z., Thompson, T. B., Zimmerman, S., Rivoire, K., Conerly, T., Olah, C., and Batson, J. On the biology of a large language model. Transformer Circuits Blog, March 27 2025. Anthropic; <https://transformer-circuits.pub/2025/attribution-graphs/biology.html>.
- Liu, C. Y., Zeng, L., Xiao, Y., He, J., Liu, J., Wang, C., Yan, R., Shen, W., Zhang, F., Xu, J., Liu, Y., and Zhou, Y. Skywork-reward-v2: Scaling preference data curation via human-ai synergy, 2025. URL <https://arxiv.org/abs/2507.01352>.
- Ma, G., Liang, Z., Chen, I. Y., and Sojoudi, S. Do sparse autoencoders identify reasoning features in language models? *arXiv preprint arXiv:2601.05679*, 2026.
- Makhzani, A. and Frey, B. K-sparse autoencoders. In *International Conference on Learning Representations*, 2014.
- Marks, S., Rager, C., Michaud, E. J., Belinkov, Y., Bau, D., and Mueller, A. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=I4e82CIDxv>.
- Mathematical Association of America. American invitational mathematics examination (aime) 2024. https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination, 2024.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.
- Mondorf, P. and Plank, B. Beyond accuracy: Evaluating the reasoning behavior of large language models - a survey. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=Lmjgl2n1lu>.
- Mu, J. and Andreas, J. Compositional explanations of neurons. *CoRR*, abs/2006.14032, 2020. URL <https://arxiv.org/abs/2006.14032>.
- Mueller, A., Brinkmann, J., Li, M. L., Marks, S., Pal, K., Prakash, N., Rager, C., Sankaranarayanan, A., Sharma, A. S., Sun, J., Todd, E., Bau, D., and Belinkov, Y. The quest for the right mediator: A history, survey, and theoretical grounding of causal interpretability. *CoRR*, abs/2408.01416, 2024. URL <https://doi.org/10.48550/arXiv.2408.01416>.
- Olah, C., Cammarata, N., Schubert, L., Goh, G., Petrov, M., and Carter, S. Zoom in: An introduction to circuits. *Distill*, 5(3), 2020. doi: 10.23915/distill.00024.001. URL <https://distill.pub/2020/circuits-zoom-in/>. Open Access.
- OpenAI. Learning to reason with llms. <https://openai.com/index/learning-to-reason-with-llms/>, 2024. Accessed: 2025-09-05.
- OpenThoughts. Open thoughts. <https://open-thoughts.ai>, 2025. Accessed: 2025-09-24.
- Plaat, A., Wong, A., Verberne, S., Broekens, J., Van Stein, N., and Back, T. Multi-step reasoning with large language models, a survey. *ACM Comput. Surv.*, November 2025. ISSN 0360-0300. doi: 10.1145/3774896. URL <https://doi.org/10.1145/3774896>. Just Accepted.

- Qwen Team. Qwq: Reflect deeply on the boundaries of the unknown. Hugging Face, 2024.
- Rai, D. and Yao, Z. An investigation of neuron activation as a unified lens to explain chain-of-thought eliciting arithmetic reasoning of LLMs. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7174–7193, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.387. URL <https://aclanthology.org/2024.acl-long.387/>.
- Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL <https://arxiv.org/abs/2311.12022>.
- Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- Sheng, L., Zhang, A., Wu, Z., Zhao, W., Shen, C., Zhang, Y., Wang, X., and Chua, T.-S. On reasoning strength planning in large reasoning models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=H26A7c191u>.
- Stolfo, A., Belinkov, Y., and Sachan, M. A mechanistic interpretation of arithmetic reasoning in language models using causal mediation analysis. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL <https://openreview.net/forum?id=aB3Hwh4UzP>.
- Sui, Y., Chuang, Y.-N., Wang, G., Zhang, J., Zhang, T., Yuan, J., Liu, H., Wen, A., Zhong, S., Zou, N., Chen, H., and Hu, X. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL <https://openreview.net/forum?id=HvoG8SxggZ>.
- Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328, 2017.
- Tang, X., Wang, X., Lv, Z., Min, Y., Zhao, W. X., Hu, B., Liu, Z., and Zhang, Z. Unlocking general long chain-of-thought reasoning capabilities of large language models via representation engineering, 2025. URL <https://arxiv.org/abs/2503.11314>.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- Wang, M., Xu, Z., Mao, S., Deng, S., Tu, Z., Chen, H., and Zhang, N. Beyond prompt engineering: Robust behavior control in LLMs via steering target atoms. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23381–23399, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1139. URL [https://aclanthology.org/2025.acl-long.1139/](https://aclanthology.org/2025.acl-long.1139).
- Wang, Y., Yu, Y., Liang, J., and He, R. A comprehensive survey on trustworthiness in reasoning with large language models. *arXiv preprint arXiv:2509.03871*, 2025b.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS ’22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.
- Wu, Z., Arora, A., Wang, Z., Geiger, A., Jurafsky, D., Manning, C. D., and Potts, C. ReFT: Representation finetuning for language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=fykjplMc0V>.
- Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., Lu, K., Xue, M., Lin, R., Liu, T., Ren, X., and Zhang, Z. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL <https://arxiv.org/abs/2409.12122>.
- Yue, X., Qu, X., Zhang, G., Fu, Y., Huang, W., Sun, H., Su, Y., and Chen, W. Mammoth: Building math generalist models through hybrid instruction tuning, 2023. URL <https://arxiv.org/abs/2309.05653>.

Zhong, J., Shen, W., Li, Y., Gao, S., Lu, H., Chen, Y., Zhang,
Y., Zhou, W., Gu, J., and Zou, L. A comprehensive survey
of reward models: Taxonomy, applications, challenges,
and future. *arXiv preprint arXiv:2504.12328*, 2025.

A. Algorithms of IPG

In this section, we provide two main algorithms of our method in order to clearly illustrate the key steps of IPG. The first algorithm focuses on *identifying reasoning-critical components* by attributing outcome-weighted signals to internal representations. The second algorithm is based on these identified components to *control reasoning behavior* of the model. (see more details in Section 3.2,3.3)

A.1. Identifying Reasoning Components

We introduce how to identify reasoning-critical components. We use a support set of M samples, which can be either a subset or the full dataset of any reasoning-related benchmarks like GSM8K (Cobbe et al., 2021). We denote output residual stream of transformer block as $\mathcal{F}(\cdot)$ which takes a sample as input and we can get the hidden state of residual stream. Here, Φ can be either the identity map, corresponding to neuron-level IPG, or the SAE encoder, corresponding to feature-level IPG. After that, we can localize the reasoning-critical components. For the calculation of attribution, we use Riemann approximation to estimate integral:

$$\text{IPG}(i; \mathbf{x}) = \sum_{t=1}^T (h_{t,i} - h'_{t,i}) \cdot \frac{1}{q} \sum_{k=1}^q \mathbb{E}_{\tau \sim \pi_\theta} \left[\frac{\partial}{\partial h_{t,i}} \log \pi_\theta \left(a_t | s_t; \mathbf{h}'_t + \frac{k}{q} (\mathbf{h}_t - \mathbf{h}'_t) \right) \cdot A^\pi(s_t, a_t) \right]. \quad (7)$$

where $\mathbf{h}' = [h'_1, \dots, h'_m]$ is a relative baseline of \mathbf{h} and q is the number of discrete steps or partitions used in the Riemann approximation to estimate the integral. \mathbf{x} is the data sample. Notably, we use \mathbf{h} in the following Algorithm 1 and Algorithm 2 to denote the representation of either neuron or SAE feature. More details about advantage A^π are provided in Section A.3.

Algorithm 1 IPG: Identifying Reasoning Components

Require: Support set $X = \{\mathbf{x}_1, \dots, \mathbf{x}_M\}$, number of components $p \in \mathbb{N}$

Ensure: Selected component indices \mathbf{P}

- 1: **for** $j = 1$ to M **do**
 - 2: Extract hidden states $\mathbf{z} = [z_1, z_2, \dots, z_n]$ from $\mathcal{F}(\mathbf{x}_j)$
 - 3: Compute components $\mathbf{h} = [h_1, h_2, \dots, h_d] = \Phi(\mathbf{z})$
 - 4: **for** $i = 1$ to m **do**
 - 5: $S(h_i; X) \leftarrow S(h_i; X) + \text{IPG}(i; \mathbf{x}_j)$ {Equation (3)}
 - 6: **end for**
 - 7: **end for**
 - 8: $\mathbf{P} \leftarrow \arg \max_{i \in [m]} S(u_i; X)$ {Equation (4)}
 - 9: **return** \mathbf{P}
-

A.2. Controlling Reasoning Behavior

After identifying reasoning-critical components, we further demonstrate how to directly control the reasoning behavior of the model. Given a set of target component indexes \mathbf{P} , we first encode the hidden state \mathbf{h} into the representation space s using Φ , which can be either the identity mapping (for neuron-level control) or the SAE encoder (for feature-level control). We then intervene on the selected components by scaling their activations with a factor γ , yielding a modified representation \mathbf{h}' . Finally, we decode s' back to the residual stream. Notably, in the SAE setting, the reconstruction also retains the error term to ensure faithful recovery of the original hidden state structure (see more details in Section B.1 about this).

A.3. Variants of IPG

In this section, we provide variants of our IPG that supports different algorithms of policy gradient (Mnih et al., 2016), as mention in Section 3.2.

In GRPO (Shao et al., 2024), the structure of IPG remains identical, except the advantage A_t is replaced with a group-relative advantage:

$$A_t = \frac{r_t - \text{mean}(\{r_1, r_2, \dots, r_N\})}{\text{std}(\{r_1, r_2, \dots, r_N\})}. \quad (8)$$

Algorithm 2 IPG: Controlling Reasoning Behavior over Generation Steps

Require: Component indices $P = \{i_1, \dots, i_p\}$, hidden states $\{\mathbf{z}_t\}_{t=1}^T$, scaling factor $\gamma \in \mathbb{R}$

Ensure: Modified hidden states $\{\mathbf{z}'_t\}_{t=1}^T$

- 1: **for** $t = 1$ to T **do**
- 2: Compute components $\mathbf{h}_t = [h_{t,1}, \dots, h_{t,d}] = \Phi(\mathbf{z}_t)$ {Equation (21)}
- 3: Initialize $\mathbf{h}'_t \leftarrow \mathbf{h}_t$
- 4: **for** $k = 1$ to p **do**
- 5: $h'_{t,i_k} \leftarrow \gamma \cdot h'_{t,i_k}$ {Scaling, Equation (5)}
- 6: **end for**
- 7: **if** Φ is Identity **then**
- 8: $\mathbf{z}_t \leftarrow \mathbf{h}'_t$
- 9: **else if** Φ is SAE-encoder **then**
- 10: $\epsilon_t \leftarrow \mathbf{z}_t - (\mathbf{W}_{\text{dec}} \mathbf{h}_t + \mathbf{b}_{\text{dec}})$
- 11: $\mathbf{z}_t \leftarrow \mathbf{W}_{\text{dec}} \mathbf{h}'_t + \mathbf{b}_{\text{dec}} + \epsilon_t$ {Equation (21)}
- 12: **end if**
- 13: **end for**
- 14: **Return** $\{\mathbf{z}'_t\}_{t=1}^T$

where T is the sequence length, N is the sample number, and π_θ is the policy parameterized by θ , i.e., the LLM to be interpreted. We treat the value of signal function $J(\cdot)$ as the reward r_t . $s_t^{(k)}$ is the prefix of the sequence k until t steps, $a_t^{(k)}$ is the next token in the sequence k , and $A^\pi(s_t, a_t)$ is the advantage function that estimates benefit of selecting a_t over the baseline, often derived from rewards like reasoning accuracy scores (Zhong et al., 2025).

$$\text{IPG}^{\text{GRPO}}(i; \mathbf{x}) = \sum_{t=1}^T (h_{t,i} - h'_{t,i}) \int_0^1 \frac{1}{N} \sum_{n=1}^N \left[\frac{\partial}{\partial h_{t,i}} \log \pi_\theta(a_t^{(n)} | s_t^{(n)}; \mathbf{h}'_t + \alpha(\mathbf{h}_t - \mathbf{h}'_t)) \cdot A_t^{\text{GRPO},(n)} \right] d\alpha. \quad (9)$$

Here, A_t denotes the advantage function, which measures how much better the chosen action a_t is compared to the expected baseline at state s_t . \mathbf{x} is the data sample. As we can see in Equation (9), our IPG can be seamlessly integrated with diverse policy gradient algorithms by integrating their advantage calculation.

For the signal function $J(\cdot)$, we support both **rule-based** and **model-based** signals. In the rule-based setting, the reward r is defined as a binary indicator of final-answer correctness:

$$J = \mathbb{I}\{\hat{y} = y\} = \begin{cases} 1, & \text{if } \hat{y} = y, \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

In contrast, model-based rewards provide a continuous evaluation of reasoning quality. For example, given a reward model $M(\cdot)$, we can assign rewards as

$$J = M(x, \hat{y}), \quad (11)$$

where M may return partial credit (e.g., proportion of correct intermediate steps) or preference scores from human-aligned models. This yields a richer supervision signal than binary correctness, enabling finer-grained attribution of reasoning ability. A comparison of these reward sources is presented in Appendix D.1.3.

A.4. Formal Justification of IPG

A.4.1. EXTENDING POLICY GRADIENT TO REPRESENTATION SPACE

In this section, we provide a rigorous justification for Equation (2), which computes the policy gradient with respect to a deterministic hidden activation \mathbf{h} rather than the policy parameters.

We consider an episodic Markov decision process and define the expected return as

$$J(\mathbf{h}) = \mathbb{E}_{\tau \sim \pi_\theta(\cdot; \mathbf{h}_t)} [R(\tau)] = \int \pi_\theta(\tau; \mathbf{h}_t) R(\tau) d\tau, \quad (12)$$

where $\tau = (s_1, a_1, \dots, s_T, a_T)$ denotes a trajectory (including s_t for cleaner proof), $R(\tau)$ is the total return, and \mathbf{h}_t is a deterministic hidden activation that influences the policy π_θ .

Assuming standard regularity conditions, we differentiate $J(\mathbf{h})$ with respect to \mathbf{h} under the integral:

$$\frac{\partial J}{\partial \mathbf{h}} = \int \frac{\partial \pi_\theta(\tau; \mathbf{h}_t)}{\partial \mathbf{h}} R(\tau) d\tau. \quad (13)$$

Applying the score-function (log-derivative) identity,

$$\frac{\partial \pi_\theta(\tau; \mathbf{h}_t)}{\partial \mathbf{h}} = \pi_\theta(\tau; \mathbf{h}_t) \frac{\partial}{\partial \mathbf{h}} \log \pi_\theta(\tau; \mathbf{h}_t), \quad (14)$$

we obtain

$$\frac{\partial J}{\partial \mathbf{h}} = \mathbb{E}_{\tau \sim \pi_\theta} \left[\frac{\partial}{\partial \mathbf{h}} \log \pi_\theta(\tau; \mathbf{h}_t) \cdot R(\tau) \right]. \quad (15)$$

Since the trajectory probability factorizes as

$$\log \pi_\theta(\tau; \mathbf{h}_t) = \sum_{t=1}^T \log \pi_\theta(a_t | s_t; \mathbf{h}_t), \quad (16)$$

the gradient can be written as

$$\frac{\partial J}{\partial \mathbf{h}} = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^T \frac{\partial}{\partial \mathbf{h}_t} \log \pi_\theta(a_t | s_t; \mathbf{h}_t) \cdot R(\tau) \right]. \quad (17)$$

For any baseline $b(s_t)$ independent of the action a_t , we have

$$\mathbb{E}_{\tau \sim \pi_\theta} \left[\frac{\partial}{\partial \mathbf{h}} \log \pi_\theta(a_t | s_t; \mathbf{h}_t) \cdot b(s_t) \right] = 0, \quad (18)$$

since $\int \pi_\theta(a_t | s_t; \mathbf{h}_t) da_t = 1$ for all \mathbf{h}_t . Subtracting this zero term yields the advantage-based form:

$$\frac{\partial J}{\partial \mathbf{h}} = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^T \frac{\partial}{\partial \mathbf{h}_t} \log \pi_\theta(a_t | s_t; \mathbf{h}_t) \cdot A^\pi(s_t, a_t) \right], \quad (19)$$

which is Equation (2).

Discussion. This derivation shows that the policy gradient formulation naturally extends to deterministic hidden activations. The gradient measures how perturbations to internal representations influence the expected return through their effect on the policy, thereby justifying the use of policy-gradient-based attribution for internal components.

A.4.2. INTEGRATE POLICY GRADIENT WITH PATH INTEGRAL

In this section, we derive the Integrated Policy Gradient (IPG). Let \mathbf{x} be the input prompt and τ be the generated trajectory with probability $\pi_\theta(\tau | \mathbf{x})$. We define the attribution for an internal representation vector h relative to a baseline h' by considering the interpolated state $h(\alpha) := h' + \alpha(h - h')$ for $\alpha \in [0, 1]$. Crucially, we define the objective function $\mathcal{J}(\alpha)$ as the *expected reward* conditioned on this interpolated state: $\mathcal{J}(\alpha) := \mathbb{E}_{\tau \sim \pi_\theta(\cdot | \mathbf{x}; h(\alpha))}[J(\tau)]$, where $J(\tau)$ is the non-differentiable metric (e.g., correctness). To attribute the change in this objective, we first derive its gradient as $\nabla_{h(\alpha)} \mathcal{J}(\alpha) = \mathbb{E}_\tau [\sum_t \nabla_{h(\alpha)} \log \pi_\theta(a_t | s_t; h(\alpha)) A_t^\pi]$. Next, we apply the fundamental theorem of calculus to the scalar function $\mathcal{J}(\alpha)$. The total difference in performance $\mathcal{J}(1) - \mathcal{J}(0)$ is given by the integral of gradients along the path:

$$\begin{aligned} \mathcal{J}(1) - \mathcal{J}(0) &= \int_0^1 \frac{d\mathcal{J}(\alpha)}{d\alpha} d\alpha = \int_0^1 \langle \nabla_{h(\alpha)} \mathcal{J}(\alpha), h - h' \rangle d\alpha \\ &= \sum_{i=1}^d (h_i - h'_i) \int_0^1 \mathbb{E}_{\tau \sim \pi_\theta(\cdot | h(\alpha))} [\nabla_{h_i(\alpha)} \log \pi_\theta(\tau | h(\alpha)) A^\pi(\tau)] d\alpha. \end{aligned} \quad (20)$$

This derivation confirms that the IPG attribution score IPG_i represents the functional contribution of the i -th component to the expected return, satisfying $\sum_i IPG_i = \mathbb{E}[J]_{\text{original}} - \mathbb{E}[J]_{\text{baseline}}$. In our experiments, Eq. (3) serves as the Monte Carlo estimator of this theoretical quantity, approximated via Riemann sums over q interpolation steps.

B. Implementation Details

The reproducible code will be released upon the acceptance of this paper.

B.1. SAE Training

k-Sparse Autoencoder Formulation. A k-Sparse Autoencoder (k-SAE) (Gao et al., 2025; Makhzani & Frey, 2014) consists of an encoder and a decoder which aim to reconstruct model activations while enforcing sparsity in the feature space. The encoding and decoding processes are shown as

$$\mathbf{f} = [f_1, f_2, \dots, f_m] = \text{TopK}(\mathbf{W}_{\text{enc}}(\mathbf{z} - \mathbf{b}_{\text{pre}})), \quad \hat{\mathbf{z}} = \mathbf{W}_{\text{dec}}\mathbf{f} + \mathbf{b}_{\text{pre}} + \epsilon(\mathbf{z}), \quad (21)$$

where \mathbf{f} has non-zero elements $|\mathbf{f}|_0 = k$ ($k < m$), $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{m \times n}$ is the encoder weight, $\mathbf{b}_{\text{pre}} \in \mathbb{R}^n$ is the bias term, and $\text{TopK}(\cdot)$ retains the top k largest values while zeroing others. $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{n \times m}$ is the decoder weight. Besides, we also add an SAE error term $\epsilon(\mathbf{z}) = \mathbf{z} - \hat{\mathbf{z}} \in \mathbb{R}^n$ to enable localized modification on residual stream.

Dataset for training k-SAEs. In this section, we provide details about training the k-SAE. Followed by (Galichin et al., 2025), we train the SAE on the activations of the model using the full *OPENTHOUGHT-114K* dataset (OpenThoughts, 2025), which is composed of high quality reasoning trace generated by DEEPSEEK-R1 including math, science, code, etc. To elicit consistent chain-of-thought style activations when harvesting activations, we apply the following chat template:

```
<system>Please reason step by step and put your answer within \boxed{}.

</system><|user|>{question}<|assistant|>{deepseek_reasoning}
{deepseek_solution}
```

Training K-SAE on Hidden Space. As shown in Equation (21), the k-SAE contains an encoder \mathbf{W}_{enc} and a decoder \mathbf{W}_{dec} initialized with the same parameters. The reconstruction loss we use is a standard mean squared error (MSE) loss defined as

$$\mathcal{L}(\mathbf{z}) = \|\mathbf{z} - \hat{\mathbf{z}}\|_2^2 \quad (22)$$

where \mathbf{z} is the original hidden state vector and $\hat{\mathbf{z}}$ denotes its reconstruction (excluding the residual error term). The dimension of the sparse representation space is 24576 for Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) and 65536 for Llama-3.1-8B-Instruct (Grattafiori et al., 2024) corresponding to an expansion factor of 16. We train k-SAE on a single NVIDIA H20 GPU, taking approximately 7 hours for Qwen and 30 hours for Llama.

Training Results. We present the details about the training of k-SAEs, including hyper-parameters and learning curves. We use the learning rate of 0.005 with batch sizes of 32, context length of 2048 tokens. For the training curve, we leverage Fraction of Variance Unexplained (FVU) (Makhzani & Frey, 2014), which is a related metric of interest, measuring the total amount of the original activation that is not “explained” or reconstructed well by k-SAE. FVU is formally defined as

$$FVU = \frac{\mathcal{L}(\mathbf{z})}{\text{var}[\mathbf{z}]} \quad (23)$$

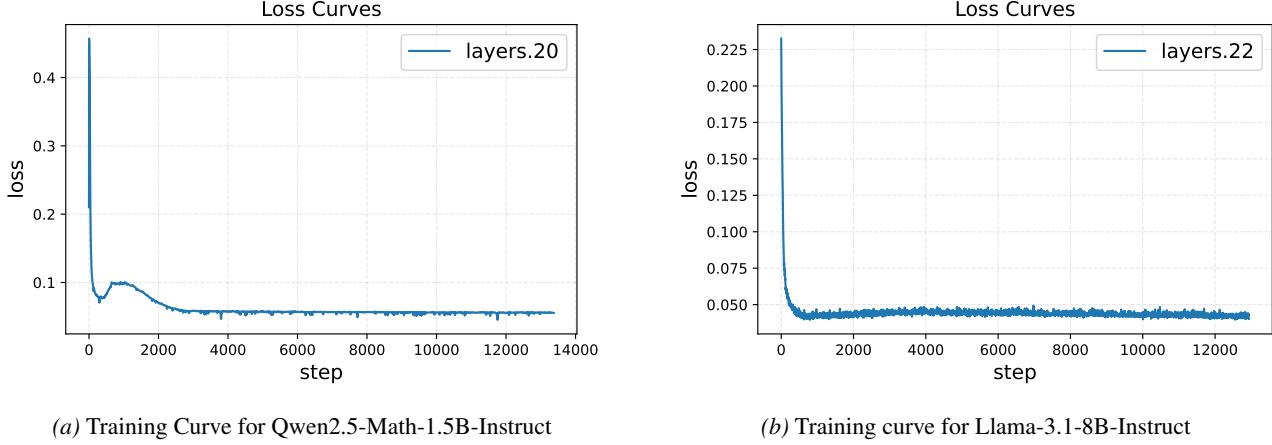
where \mathbf{z} is the hidden state, $\mathcal{L}(\mathbf{z})$ is defined in Equation (22) and var represents the variance of \mathbf{z} . A lower FVU indicates better reconstruction performance since more original activation is captured by the k-SAE model. The training curves in Figure 7 show that the k-SAE perform well on the FVU metric.

Discussion: The Role of SAE Error Node. While the k-SAE can effectively reconstruct the hidden state $\hat{\mathbf{h}}$, residual gaps inevitably remain. Recent studies (Lindsey et al., 2025; Marks et al., 2025) highlight that explicitly modeling error nodes is essential to fill these gaps, as it gives a principled decomposition of model behaviors into contributions from interpretable features and error components not yet captured by our SAEs. In our setting, we retain the SAE error term to preserve the reliability of our downstream intervention, thus ensuring that our modification is incremental without degrading the model’s performance.

B.2. IPG Attribution Details

Hyperparameter for Attribution.

This section summarizes the hyperparameters used for the attribution method throughout our experiments. Unless otherwise specified, the same configuration is applied across all datasets and models.



(a) Training Curve for Qwen2.5-Math-1.5B-Instruct

(b) Training curve for Llama-3.1-8B-Instruct

Figure 7. K-SAE Training Curve

Table 4. IPG-GRPO Hyperparameter Settings.

Attribution Configuration	
Path integral steps	10
GRPO group size	3,5
Decoding Strategy	
Top- <i>p</i>	0.8
Temperature	0.6
Max Generation Tokens	512,2048,8192

B.3. Fine-grained Reasoning Mechanism Discovery

B.3.1. ASPECT-LEVEL GRANULARITY:

In this section, we provide more implementation details on fine-grained reasoning mechanism discovery. Specifically, we rank reasoning components by their IPG scores and automatically label the top 30 neurons and SAE features identified from Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) using a GPT-5-mini API respectively. For interpretation, we manually decompose the comprehensive reasoning into four hierarchical layers in reasoning: *Semantic Comprehension*, *Problem Decomposition*, *Reasoning Depth* and *Numerical Accuracy*. Importantly, our analysis is causal: we directly intervene on each neuron or feature, collect all newly-correct answers after steering, and compare pre- and post-intervention responses. The resulting behavioral changes are then judged by the API to correspond to one of the four reasoning abilities. These categories align with the labeling scheme defined in the system prompt used for API calls, which is shown below.

```
You are a careful evaluator. Output JSON only (an object). Do not include any extra text.
Allowed labels (with concise role notes):
1. FineGrainedSemanticComprehension [Semantic Layer]
Evaluates precise understanding of wording, references, modifiers, units.
Focus: resolving ambiguity/negation/quantifiers so the problem is interpreted correctly.
2. ProblemDecompositionAndLogicalSequencing [Planning Layer]
Evaluates whether the task is decomposed into necessary subgoals and ordered coherently.
Focus: a plan that covers all required steps and aligns with the objective.
3. ReasoningDepthAndThoroughness [Logical Execution Layer]
Evaluates completeness and correct application of rules/conditions during non-numerical reasoning.
Focus: ensuring all relevant conditions are correctly applied during each step, variable states are continuously tracked, and implicit logical premises are not omitted.
4. NumericalAccuracyAndCalculation [Calculation Layer]
Evaluates correctness of arithmetic/algebra/probability and unit conversions.
Focus: operation order, numeric consistency, and appropriate rounding.
```

B.3.2. PROCESS-LEVEL GRANULARITY:

We analyze reasoning behavior at the process level by characterizing how internal components are engaged over the course of multi-step inference.

Activation Frequency. For a given component (neuron or SAE feature) and a reasoning segment, we define the *activation frequency* as the proportion of tokens within the segment for which the component is active. Formally, let \mathcal{T} denote the set of tokens belonging to a reasoning segment, and let $\mathbb{I}(f_t > 0)$ be an indicator function that equals 1 if the feature activation at token t is non-zero (i.e., selected by the SAE), and 0 otherwise. The activation frequency is defined as

$$\text{Freq}(f) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \mathbb{I}(f_t > 0). \quad (24)$$

This normalization by the segment length ensures that activation frequency is not confounded by variations in token count across reasoning steps or problems. All statistics are first computed per example and then averaged across the dataset.

Reasoning Step Delimitation. To obtain discrete reasoning steps, we segment each chain-of-thought by line breaks, treating consecutive text blocks separated by the delimiter “\n\n” as individual reasoning steps. This heuristic follows common formatting conventions in chain-of-thought reasoning, where each paragraph typically corresponds to a coherent sub-step such as problem decomposition, intermediate calculation, or verification.

Step Deciles. As different problems may contain varying numbers of reasoning steps, we normalize the reasoning progress of each example to the interval [0, 1]. Specifically, for a chain-of-thought consisting of S reasoning steps, the i -th step is assigned a normalized progress value $(i - 1)/(S - 1)$. We then discretize the normalized reasoning progress into 8 equal-width bins, referred to as *reasoning step deciles*. For each decile, activation frequency statistics are aggregated across all reasoning steps whose normalized progress falls within the corresponding bin. This procedure enables alignment and comparison of temporal activation patterns across examples with different reasoning lengths.

C. Experiment Settings

In this section, we provide details on the experiment settings (Section 4.1), including our evaluation pipeline and the baseline implementation.

C.1. Evaluation Settings

For all benchmarks, we employ the lm-evaluation-harness framework (Gao et al., 2024) as the evaluation tool. For Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024), we set the maximum tokens to 2048, while for Llama-3.1-8B-Instruct (Grattafiori et al., 2024), we set the maximum new tokens to 8192. Followed by prior work (Dutta et al., 2024), interventions with IPG are applied to the mid-to-late transformer blocks, specifically layers 20–26 for Qwen and layers 22–28 for Llama. To avoid randomness and ensure reproducibility, we set do sample to false, which is equivalent to greedy decoding strategy (temperature set as 0). Accordingly, we employ Accuracy@1 as our metric.

C.2. Baselines

Random. For comparison, we construct a random baseline by uniformly and randomly selecting neurons within the hidden states \mathbf{h} and applying interventions based on these neurons. The final results of this baseline are averaged across multiple random seeds.

Activation. Internal neuron activations contain meaningful information. Additionally, (Dai et al., 2022) demonstrates the ability to edit internal neurons to influence neural network performance. Building on this, we adopt a similar approach, performing intervention directly on the original activations in the hidden state. For a fair comparison, we constrain our intervention to the residual stream of a language model.

Reasoning Neuron. As described in Rai & Yao (2024), the focus is on analyzing activation patterns, such as average activation values in the feed-forward (FF) layer, to identify reasoning neurons and interpret them using predefined concepts C , such as arithmetic operations C_{add} with key tokens like “add”, “+”. We extend this approach to operate on the residual stream of transformer blocks (Vaswani et al., 2017). By leveraging all provided concepts, we identify related neurons and use them for intervention.

Control Vector. Introduced by Højer et al. (2025), it employs representation engineering by extracting model activations from the residual stream of a large language model (LLM) during a reasoning task. These activations are used to derive a control vector for inference-time intervention. The control vector is constructed from positive and negative pairs: positive pairs consist of Chain-of-Thought (CoT) prompts where the model produces correct outputs, while negative pairs comprise 75 random character strings sampled from the alphabet (A–Z). Then PCA is applied and the first principal component in the control direction is treated as the most reasoning-related direction. Following the open-source repository, we use 30% of the samples from each dataset to compute the control vector.

SAE-Reasoning. As introduced by Galichin et al. (2025), it employ Sparse Autoencoders (SAEs) (Makhzani & Frey, 2014; Gao et al., 2025) and learn a sparse decomposition of latent representations of a neural network into interpretable features, to identify features that drive reasoning in the DeepSeek-R1 (Guo et al., 2025) series of models. They extract candidate “reasoning feature” from SAE representations, that based on specific tokens such as {“wait”, “but”}, associating feature activation with these tokens. For each feature, *reasonscore* is computed based on its feature activation on the OPENTHOUGHT-114K (OpenThoughts, 2025). After that, following the releasing implementation, we intervene the feature with top *reasonscore*.

D. More results

In this part, we present more results of our experiments part and studies for better illustrations of our IPG.

D.1. Sensitive Study

D.1.1. K-SAE HYPERPARAMETER

We adjust different k values to evaluate the effects of the k-SAEs (Gao et al., 2025) on reasoning behavior control as shown in Table 5. The results are based on Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) layer 20.

GSM8K		
k	Acc. (Enhance) \uparrow	Acc. (Suppress) \downarrow
Original	82.41	82.41
8	84.38	82.26
32	84.76	81.73
64	83.70	81.96

Table 5. Investigation of the impact of varying TopK parameters in k-SAE (Belitsky et al., 2025) on controlling reasoning behavior in Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) for GSM8K (Cobbe et al., 2021). TopK refers to preserving the top k features while deactivating others in the feature space, as described in Section 3.3.

In Table 5, as the TopK parameter k increases, the ability to control reasoning behavior, including both enhancing and suppressing, either persists or begins to diminish. We demonstrate that our results are not heavily dependent on the choice of Sparse Autoencoders (SAEs). Instead, our focus is on the accurate identification and intervention of reasoning features. Consequently, we select $k = 32$ for the k-SAE (Gao et al., 2025) across all experiments.

D.1.2. IPG ATTRIBUTION HYPERPARAMETER

We conduct experiment on the effects of different number p of components identified, as stated in Section 3.2. In Table 6, as the number of top- p neurons increases, the control over reasoning behavior becomes more effective. However, beyond a certain point, adding more neurons does not yield further improvements. This is because reasoning-related concepts may be distributed across multiple neurons, and the top neurons already exert significant control, while additional neurons lack substantial reasoning-related functionality.

p	GSM8K	
	Acc. (Enhance) \uparrow	Acc. (Suppress) \downarrow
Original	82.41	82.41
10	82.94	76.72
20	83.85	73.24
30	83.78	72.71
40	83.85	82.18
50	84.60	10.69
60	84.15	81.20

Table 6. Evaluation on the impact of varying the top- p number of neurons in 3.3 in Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) for GSM8K (Cobbe et al., 2021). This evaluation is based on our k-SAE implementation (Gao et al., 2025) with a fixed $k = 32$.

# Samples	GSM8K	
	Acc. (Enhance) \uparrow	Acc. (Suppress) \downarrow
1	82.79	82.03
10	83.17	79.37
20	83.17	6.52
50	83.54	12.81
100	83.39	12.81
500	83.55	12.05

Table 7. Effect of batch size (# samples used to compute IPG directions) on steering performance. Model: Qwen2.5-Math-1.5B-Instruct on GSM8K (Cobbe et al., 2021). Original accuracy: 82.41%.

We also conduct sensitivity analysis on our choice of batch size M . We perform an ablation on the number of samples used to compute the IPG directions, varying it from 1 to 500. All other settings follow the main experimental protocol (Table 7): Qwen2.5-Math-1.5B-Instruct (Qwen Team, 2024) evaluated on GSM8K (Cobbe et al., 2021) under greedy decoding. “Enhance” reports accuracy after strengthening the identified reasoning components (higher is better), while “Suppress” reports accuracy after weakening them (lower is better). The original accuracy of the model is 82.41%.

When M is very small (≤ 10), the gradient estimates exhibit high variance, which degrades component identification and leads to weak performance. As it increases to ≥ 20 , variance decreases rapidly, yielding smoother and more reliable gradient signals. This stabilizes both enhancement and suppression, with suppression improving dramatically.

D.1.3. REWARD SIGNALS

Model	Reward Method	GSM8K	MATH-500	AIME-2024	GPQA Diamond
Qwen2.5-Math-1.5B	Rule Fuction	84.15	64.20	20.00	26.77
	Reward Model	84.60	65.80	16.67	33.33

Table 8. Performance of IPG in enhancing accuracy across reasoning datasets for Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) under different reward signals.

We report the performance of IPG in improving accuracy across reasoning datasets for Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) using different reward signals, including rule-based and reward model-based signals, as detailed in Appendix A.3. The scaling intervention (see Section 3.3) is applied at the 24th layer. As shown in Table 8, reward model-based signals outperform rule-based signals in most datasets, suggesting that reward models capture richer information or better represent reasoning outcomes, thereby enhancing IPG’s effectiveness in identifying and controlling reasoning behavior.

D.1.4. BASELINE CHOICE.

To further verify the robustness of IPG, we conduct additional experiments on different choices of the baseline hidden state h' using Qwen-Math-2.5-1.5B-Instruct (Qwen Team, 2024) on the GSM8K (Cobbe et al., 2021) dataset. We report accuracy under greedy decoding as described in Section 4.1. The original accuracy of the model is 82.41%. We compare

h'	GSM8K	
	Acc. (Enhance) \uparrow	Acc. (Suppress) \downarrow
Zero vector	84.60	10.69
Mean hidden state	84.45	21.15
Random noise	84.00	78.16

Table 9. Sensitivity analysis on the choice of baseline representation h' . The model is Qwen-Math-2.5-1.5B-Instruct (Qwen Team, 2024) evaluated on GSM8K (Cobbe et al., 2021). Original accuracy before any intervention: 82.41%.

three baseline strategies: (i) the zero vector, (ii) the mean hidden state computed over the training set, and (iii) random Gaussian noise $\mathcal{N}(0, I)$ as an explicit non-informative reference.

Results are shown in Table 9. Using the zero vector or the mean hidden state yields nearly identical enhancement performance (84.45% and 84.60%, respectively) and both enable almost perfect suppression (10.69% and 21.15%). In contrast, random noise performs considerably worse on suppression (78.16%), confirming that it does not constitute a meaningful baseline representation. These findings demonstrate that IPG is highly robust to the specific choice of h' as long as it represents a reasonable direction; arbitrary noise fails to serve this role effectively.

D.2. Controlling Reasoning Length

Beyond identifying reasoning components using accuracy-based reward signals, we explore whether neurons controlling the length of reasoning can be identified. This is achieved by modifying the rule function (see Appendix A.3) to prioritize sensitivity to reasoning length rather than accuracy. Specifically, the rule function is redefined to reward outputs based on their generated length, encouraging the identification of neurons that influence the verbosity or conciseness of reasoning.

The modified signal function, $J(\cdot)$, is defined as a function of the generated text length, $|\tau|$, measured as the number of tokens in the output, τ is the generated trajectory. The reward increases with length to promote longer reasoning chains or decreases to favor conciseness, depending on the desired behavior. The rule function is given by

$$J(\tau) = f(|\tau|), \quad (25)$$

where $f(\cdot)$ can be any monotonically increasing function. For example, we can have a 2-order function $f(L) = \alpha L^2 + \beta L + \gamma$ where α , β , and γ are tunable parameters that control the reward's sensitivity to length.

As shown in Figure 3 in Section 4.2, scaling the top one percent of neurons identified via length-based rewards on DeepSeek-Qwen-1.5B (Guo et al., 2025) leads to a gradual decrease in the model's output length as the scaling factor increases. In conclusion, these results demonstrate that our IPG can be readily extended to capture diverse outcome-aware signals corresponding to specific behaviors, highlighting its flexibility and effectiveness.

D.3. Ablation on Policy Gradient

In the main paper, we accumulate policy gradients along a continuous path from a baseline activation h' to the original activation h , yielding *Integrated Policy Gradients* (IPG). This path-integral formulation produces baseline-aware attribution scores that reduce gradient noise and improve faithfulness. In this section, we provide an ablation study comparing IPG against *raw policy gradient* (PG) attribution.

Experimental Setup. We conduct experiments on QWEN2.5-MATH-1.5B-INSTRUCT (Yang et al., 2024) using two mathematical reasoning benchmarks: MATH500 (Hendrycks et al., 2021) and AIME2024 (Mathematical Association of America, 2024). We report accuracy (%). For each dataset, we evaluate five settings: the original model without intervention, PG-based attribution (with and without SAE), and IPG-based attribution (with and without SAE). Importantly, all experimental settings share identical implementations and hyperparameters; the only difference lies in the attribution signal, i.e., raw policy gradients versus integrated policy gradients.

Results. Table 10 summarizes the results. Raw policy gradient attribution provides only modest improvements over the original model, and its effectiveness is limited even when combined with SAE. In contrast, IPG consistently yields larger performance gains across both datasets, with or without SAE. Notably, IPG achieves substantial improvements on the more challenging AIME2024 benchmark, highlighting its robustness in difficult reasoning scenarios.

Table 10. Ablation study comparing raw policy gradient (PG) and integrated policy gradient (IPG) attribution on QWEN2.5-MATH-1.5B-INSTRUCT. Accuracy (%) is reported.

Method	MATH500 \uparrow	AIME2024 \uparrow
Original	63.00	10.00
PG	64.60	16.67
IPG	65.80	20.00
PG-SAE	64.40	16.67
IPG-SAE	66.00	20.00

Analysis and Conclusion. These results demonstrate that the gains of IPG are not merely due to policy gradient attribution itself, but critically depend on the *integration along the activation path*. By aggregating gradients across activation levels, IPG mitigates local gradient noise and captures sustained, cumulative contributions of internal components. This leads to more reliable identification of influentially relevant components and, consequently, more effective intervention. Overall, this ablation confirms that the *integrated* design is a key factor underlying the superior performance of IPG.

D.4. General Reasoning Mechanism Identification

Metric Definition. We quantify consistency using the Dice similarity coefficient. Formally, let \mathbf{P}_A and \mathbf{P}_B denote the sets of neuron indices identified by a method in datasets A and B , respectively. The similarity score is defined as:

$$\text{Dice}(\mathbf{P}_A, \mathbf{P}_B) = \frac{2|\mathbf{P}_A \cap \mathbf{P}_B|}{|\mathbf{P}_A| + |\mathbf{P}_B|}. \quad (26)$$

A higher score indicates that the method identifies a similar set of critical components regardless of the input domain.

Quantitative Results. Figure 8 presents the pairwise similarity matrix. The matrix is asymmetric to facilitate a direct comparison between our method and a random baseline:

- **Lower Triangular (Proposed IPG):** These elements display the overlap of neuron indices discovered by our method. We observe a high degree of consistency across all dataset pairs, with Dice scores ranging from 0.50 to 0.69. Notably, the overlap between GSM8K and MATH500 is particularly strong (0.69), suggesting that the core reasoning mechanisms for grade-school and competition-level math are highly aligned. Even when comparing vastly different distributions, such as GSM8K and GPQA, the similarity remains robust at 0.50.
- **Upper Triangular (Random Baseline):** To contextualize these results, the upper triangular elements report the Dice similarity for a random baseline (RND.), where neuron indices are selected uniformly at random. The resulting scores are consistently low, ranging between 0.08 and 0.14.

Conclusion. The significant margin between the lower and upper triangular values confirms that IPG captures systematic, task-agnostic reasoning structures. The high overlap scores (> 0.50) compared to the random baseline (≈ 0.10) demonstrate that the identified internal mechanisms are stable and transferable across varying levels of problem complexity.

D.5. Discussion

D.5.1. CAUSALITY-MOTIVATION OF IPG

While standard gradient-based attribution method measures local sensitivity, we argue that our IPG framework more *causality-motivated* than correlation-based method (eg. SAE-R) for two reasons. We list the comparison in Table 11.

CMA-style Attribution: IPG is conceptually aligned for Causal Mediation Analysis (Mueller et al., 2024). Inspired by CMA, we treat neuron/SAE feature sets as high-dimensional mediators and quantify their contributions to the final reasoning outcome. Instead of computationally prohibitive brute-force patching, IPG uses path-integrated gradients to propose high-quality candidates for causal mediators.

Causal Validation via Intervention: Crucially, we do not rely on attribution alone. The causal link is established in Section

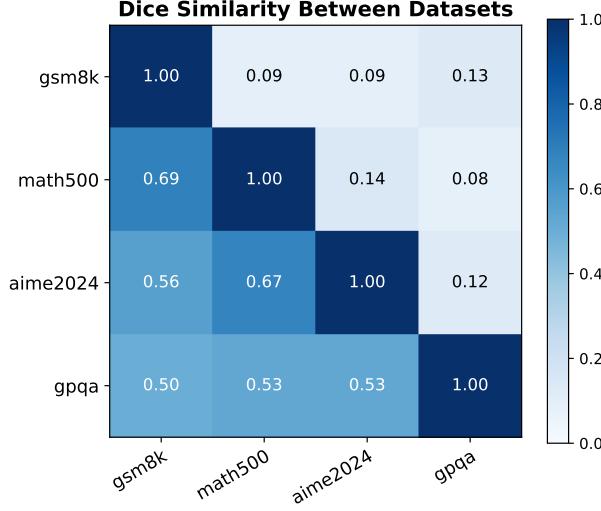


Figure 8. Dice similarity between neuron indices across datasets. We compare neuron indices discovered in 4 different datasets, highlighting shared reasoning structure across tasks, with diagonal showing random baseline.

Table 11. What IPG Provides Compared to Correlation-Based Methods

Aspect	Correlation-Based (e.g., SAE-R)	IPG
Signal source	Text pattern co-occurrence	Reasoning outcome (accuracy/length)
Attribution target	Activates with reasoning tokens	Influences final outcome
Temporal scope	Local / point-wise	Long-horizon (full trajectory)

3.3. We empirically validate causal roles of identified components underlying the reasoning mechanism subsequently in Section 4.2.

D.5.2. EFFICIENCY OF IPG

Table 12. Comparison of Inference Time (in seconds per sample) for Rule-based and Reward Model-based reward signals on GSM8K (Cobbe et al., 2021) across Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) and Llama3.1-8B-Instruct (Touvron et al., 2023)

Model	Reward Signal	Attribution Time (s/sample)
Qwen	Rule	11.6
Qwen	Reward Model	13.5
Llama	Rule	18.25
Llama	Reward Model	20.0

The table presents a comparison of attribution times (see in Section 3.2) in seconds per sample for the GSM8K (Cobbe et al., 2021) dataset using rule-based and reward model-based methods across two large language models, Qwen2.5-Math-1.5B-Instruct (Yang et al., 2024) and Llama3.1-8B-Instruct (Touvron et al., 2023). We can see in Table 12 that our IPG requires few time for attribution. This efficiency highlights the scalability of IPG, enabling rapid identification of key neurons or features in reasoning tasks, making it a practical choice for further controlling large-scale language models.

D.5.3. LIMITATION OF IPG

Reward Signal: Our current implementation of IPG relies on relatively preliminary reward definitions, including either a rule-based signal or a lightweight reward model. Such signals can be coarse and fail to capture subtle reasoning improvements. Consequently, they may introduce noise or bias when attributing causal contributions. More sophisticated approaches, such as process reward models (PRMs) or step-level verifiers, could provide finer-grained supervision that

better aligns with reasoning quality. Exploring such enhanced reward sources is a promising direction for future work.

Selection of Scaling Factor: In the present IPG implementation, the scaling factor (γ) is applied uniformly across neurons or features. In spite of its general intervention effects, it neglects more delicate things where different components may require different intervention strengths. A promising future work is to adopt component-specific scaling, allowing each neuron or feature to be steered by an individually optimized factor.

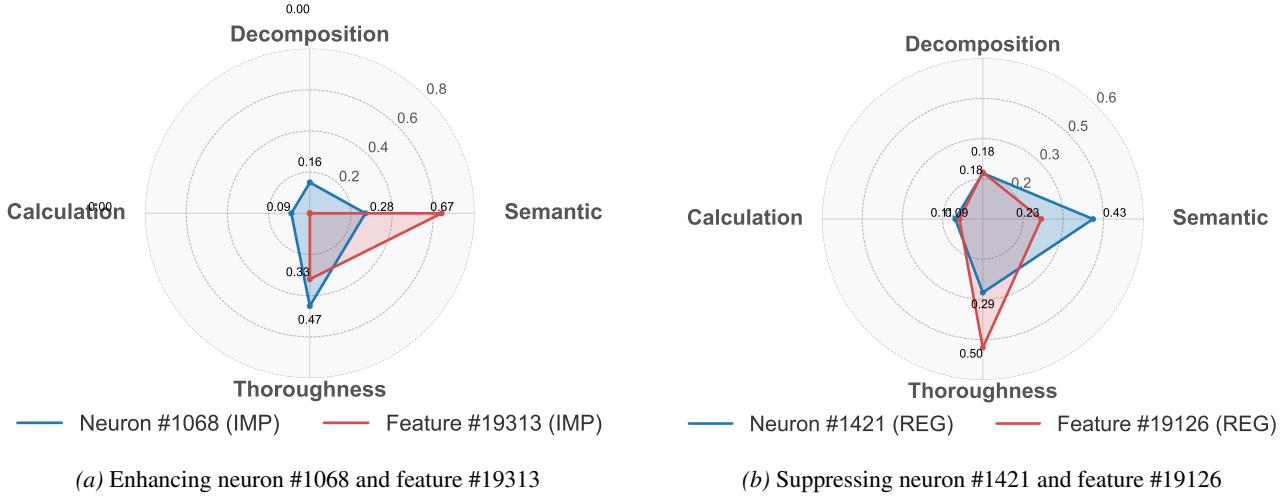
D.6. Case Studies

D.6.1. GRANULAR REASONING ASPECTS CONTROL

In this section, we provide more details about radar graphs of neuron and feature's granular reasoning aspects mentioned in Section 4.4.

Table 13. Clusters of reasoning ability type obtained using GPT-5 mini. Each cluster corresponds to a reasoning type, the representative neuron index, and the behavior changes observed under neuron steering. Enhancing amplifies desired reasoning ability, while suppressing degrades it.

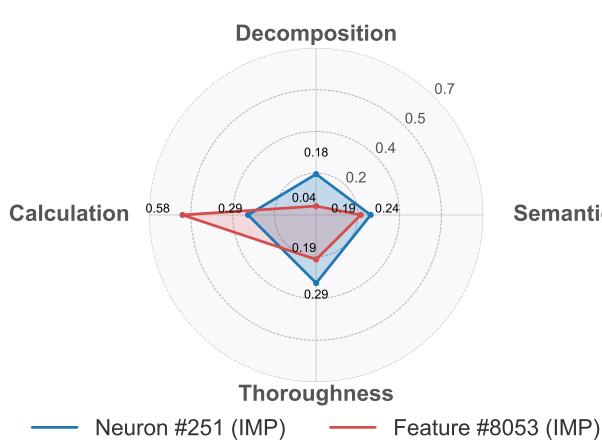
Cluster Type	Neuron Index	Feature Index	Behavior Change (Enhance / Suppress)
Fine-Grained Semantic Comprehension	520, 609, 940	19313, 19156	<i>Enhance:</i> More precise semantic parsing <i>Suppress:</i> Ignoring key constraints and details
Problem Decomposition & Structuring	553, 802, 904	19126, 20154	<i>Enhance:</i> More coherent planning of solution steps <i>Suppress:</i> Fragmented or illogical solution planning
Reasoning Depth & Thoroughness	609,940,1068	1083,18195,19126	<i>Enhance:</i> More rigorous application of conditions <i>Suppress:</i> Failure to track changing states
Numerical Accuracy & Calculation	335, 520, 553	779, 8053, 10508	<i>Enhance:</i> More precise execution of calculations <i>Suppress:</i> Increased frequency of arithmetic errors



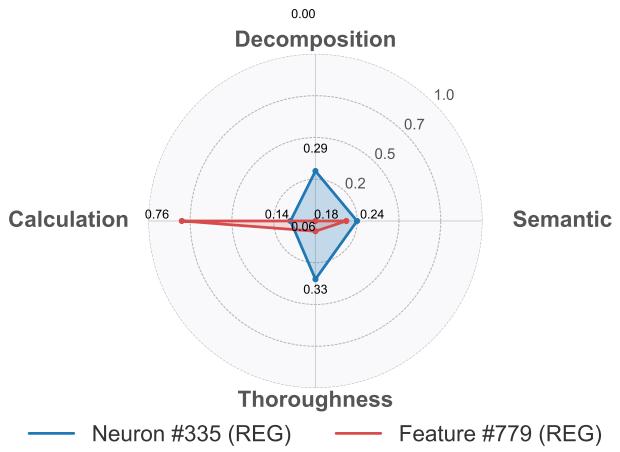
Experiment Setup:

D.6.2. GENERATED RESPONSES BEFORE AND AFTER INTERVENTION

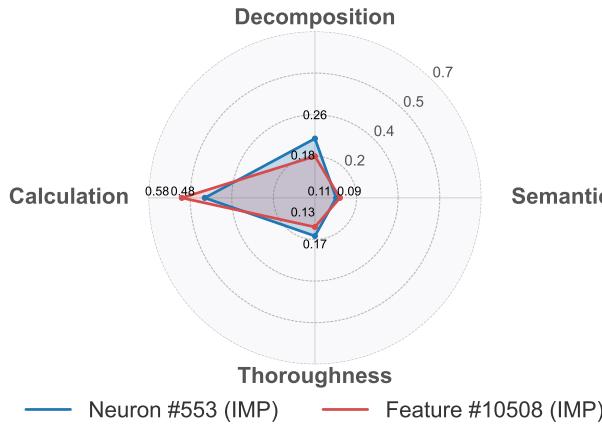
In this section, we provide more examples with our IPG framework corresponding to Section 3.3. We present generated responses before and after intervention on multiple models, spanning different reasoning benchmarks and intervention granularities (neuron-level vs. SAE feature-level). Across models and tasks, we can consistently observe that steering IPG-identified components leads to more coherent intermediate reasoning steps and improved final answers.



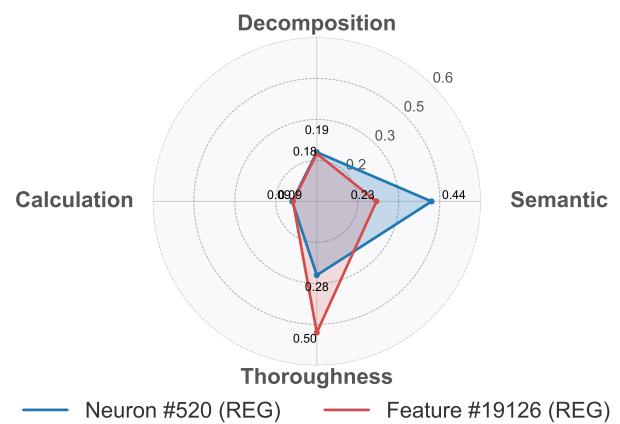
(a) Enhancing neuron #251 and feature #8053



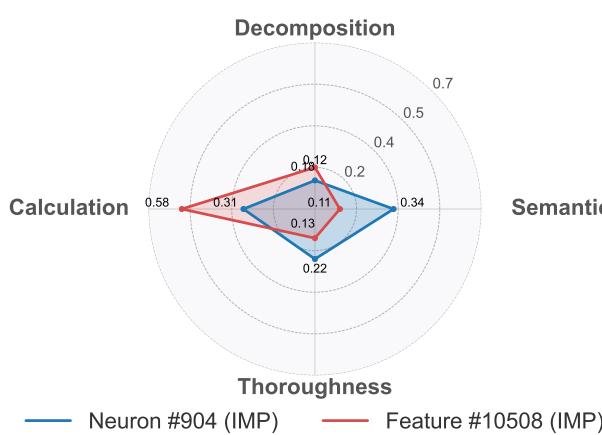
(b) Suppressing neuron #335 and feature #779



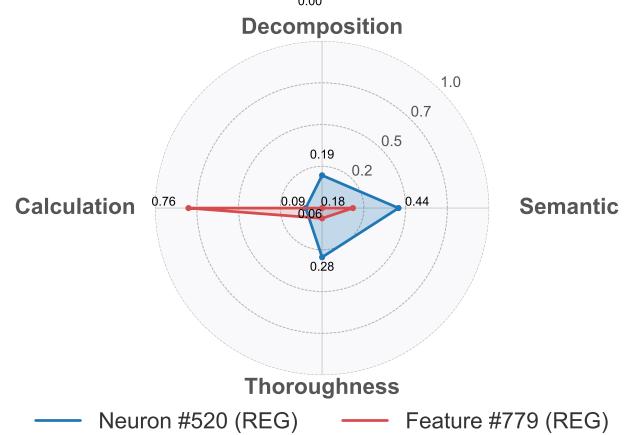
(a) Enhancing neuron #553 and feature #10508



(b) Suppressing neuron #520 and feature #19126



(a) Enhancing neuron #553 and feature #10508



(b) Suppressing neuron #520 and feature #779

Question

How many r are there in the word strawberry?

Response

Original:

To solve this problem, I need to count the number of letters in the word \strawberry." Okay, so first, I'll start by writing down the word: strawberry. Let me see, that's an 8-letter word. Hmm, wait, let me count each letter to be sure. Starting from the beginning: s, t, r, a, w, b, r, e, r. So that's s (1), t (2), r (3), a (4), w (5), b (6), r (7), e (8), r (9). Wait, that's nine letters. But I thought it was an 8-letter word. Maybe I'm miscounting. Wait, let me check again. Maybe I missed a letter. So, \strawberry" is spelled S-T-R-A-W-B-E-R-R-Y. Wait, that's 9 letters. Hmm, maybe I'm getting confused. Let me write it out: S (1), T (2), R (3), A (4), W (5), B (6), E (7), R (8), R (9), Y (10). Wait, that's 10 letters. Now I'm really confused. Is it 10 letters? Or is it 9? Wait, let me look it up. Wait, no, I can't look it up, I need to count. Okay, let's try again. S-T-R-A-W-B-E-R-R-Y. So S is 1, T is 2, R is 3, A is 4, W is 5, B is 6, E is 7, R is 8, R is 9, Y is 10. So it's 10 letters. But I thought it was an 8-letter word. Wait, maybe it's a different word. Wait, maybe the word is \strawberry" without the 'y'? Let me check. No, the spelling is S-T-R-A-W-B-E-R-R-Y, which is 10 letters. So the problem must be about the letter 'r' instead of the word. So the question is, how many 'r's are there in the word \strawberry." So if the word is 10 letters, how many 'r's are there? Wait, let me count again. S-T-R-A-W-B-E-R-R-Y. So the letters are: 1: S 2: T 3: R 4: A 5: W 6: B 7: E 8: R 9: R 10: Y. So there are two 'R's: one at position 3 and another at position 8. So the answer should be 2.

</think>

2

Enhanced:

Okay, so I need to figure out how many "r"s are in the word "strawberry." Let me start by writing down the word: S-T-R-A-W-B-E-R-R-Y. Hmm, that's a bit long, so I should count each "r" individually. First, I notice the letters S, T, R, A, W, B, E, R, R, Y. Let's break them down one by one. The first R is after S and T, so that's one "r." Then, moving along, the next R is right before E, so that's another "r." After that R, there are two more R's before Y. So let me count them:
 1. R after S 2. R after T 3. R after A 4. R after B 5. R after E 6. R before Y
 Wait, that's six R's. Let me make sure I didn't miss any. Let me go through each letter again: 1. S: Not an R. 2. T: Not an R. 3. R: 1 4. A: Not an R. 5. W: Not an R. 6. B: Not an R. 7. E: Not an R. 8. R: 2 9. R: 3 10. Y: Not an R.
 So yeah, only three R's. Wait, that contradicts my previous count. Hmm, maybe I miscounted earlier. Let me recount more carefully.
 Let me list each letter and note if it's an R:
 1. S: No 2. T: No 3. R: Yes (1) 4. A: No 5. W: No 6. B: No 7. E: No 8. R: Yes (2) 9. R: Yes (3) 10. Y: No
 Ah, so only three R's. So the answer should be 3. I think I confused myself earlier, but now it's clear. So the total number of "r"s in "strawberry" is 3.

</think>

3

Case Study on Deepseek-R1-Distilled-Qwen1.5B . Steering one neuron with highest IPG score with scaling factor $\gamma = 1.1$ on layer 20.

Question

Ellen decided to play a prank on her friend. She got a case of 12 sodas and shook 3 of them up. Then she took 1 unshaken soda for herself and left. Ellen's brother stopped by and took 1 of the shaken sodas and 2 of the unshaken sodas, then Ellen's friend came along. What is the likelihood, expressed as a percentage, that Ellen's friend gets sprayed with soda from a shaken can?

Response
Original:

To determine the likelihood that Ellen's friend gets sprayed with soda from a shake can, we need to follow the sequence of events and calculate the number of sodas in each state.

Initial number of sodas: Ellen starts with 12 sodas.

Sodas after Ellen shakes 3: Ellen shakes 3 sodas, so there are $(12 - 3 = 9)$ sodas left.

Sodas after Ellen takes 1 unshaken soda: Ellen takes 1 unshaken soda, so there are $(9 - 1 = 8)$ sodas left, with 3 sodas still shaken.

Sodas after Ellen's brother takes 1 shake soda and 2 unshaken sodas: Ellen's brother takes 1 shake soda and 2 unshaken sodas, so there are $(8 - 1 - 2 = 5)$ sodas left, with $(3 - 1 = 2)$ sodas still shaken.

Sodas after Ellen's friend comes along: Ellen's friend now has the remaining 5 sodas, with 2 of them being shake sodas.

The probability that Ellen's friend gets sprayed with soda from a shake can is the ratio of the number of shake sodas to the total number of sodas Ellen's friend has. This probability is calculated as follows:

$$\text{Probability} = \frac{\text{Number of shake sodas}}{\text{Total number of sodas}} = \frac{2}{5}$$

To express this probability as a percentage, we multiply by 100:

$$0.4 \times 100 = 40\%$$

Thus, the likelihood that Ellen's friend gets sprayed with soda from a shake can is $\boxed{40\%}$.

Enhanced:

To determine the likelihood that Ellen's friend gets sprayed with soda from a shake can, we need to follow the sequence of events and calculate the number of sodas in each state.

Initial number of sodas: Ellen starts with 12 sodas.

Sodas after Ellen shakes 3: Ellen shakes 3 sodas, so she has 9 sodas left unshaken.

Unshaken sodas = $12 - 3 = 9$

Shaken sodas = 3

Sodas after Ellen takes 1 unshaken soda: Ellen takes 1 unshaken soda for herself, so she has 8 unshaken sodas left.

Unshaken sodas = $9 - 1 = 8$

Shaken sodas = 3

Sodas after Ellen's brother takes 1 shake and 2 unshake: Ellen's brother takes 1 shake soda and 2 unshake sodas, so the number of sodas changes as follows:

Shake sodas = $3 - 1 = 2$

Unshake sodas = $8 - 2 = 6$

Sodas after Ellen's friend comes along: Ellen's friend comes along and gets 1 soda. The probability that this soda is from a shake can is the number of shake sodas divided by the total number of sodas left.

Total sodas left = $2 + 6 = 8$

Probability that Ellen's friend gets a shake soda = $\frac{2}{8} = 0.25$

Percentage = $0.25 \times 100 = 25\%$

Therefore, the likelihood that Ellen's friend gets sprayed with soda from a shake can is $\boxed{25\%}$.

Case Study on GSM8K for Qwen2.5-Math-1.5B-Instruct. Steering 5% IPG neurons with scaling factor $\gamma = 1.1$ on layer 20.

Question

John wins an award at work. The award has a 1 time monetary reward of \$5000 and a 5% raise in salary. If he makes \$2000 a week, every week of the year and he got the award on the first day of the year how much money did he make that year?

Response

Original:

To determine how much money John made that year, we need to consider both his regular weekly earnings and the additional monetary reward he received from the award.
 Calculate John's regular annual earnings:
 John makes \$2000 per week. There are 52 weeks in a year.
 $\text{Regular annual earnings} = 2000 \times 52 = 104000$
 Calculate the monetary reward from the award:
 The award has a 1 time monetary reward of \$5000.
 Calculate the total earnings for the year:
 Add the regular annual earnings to the monetary reward.
 $\text{Total earnings} = 104000 + 5000 = 109000$
 Therefore, the total amount of money John made that year is $\boxed{109000}$.

Enhanced:

To determine how much money John made that year, we need to consider both his regular weekly earnings and the additional monetary reward he received from the award.
 Calculate John's regular annual earnings:
 John makes \$2000 per week. There are 52 weeks in a year.
 $\text{Regular annual earnings} = 2000 \times 52 = 104000$
 Calculate the monetary reward from the award:
 The award has a 1 time monetary reward of \$5000.
 Calculate the 5% raise in salary:
 The 5% raise in salary is applied to his regular weekly earnings.
 $\text{Raise per week} = 2000 \times 0.05 = 100$
 Since the raise is applied to each week, the total raise for the year is:
 $\text{Total raise for the year} = 100 \times 52 = 5200$
 Calculate the total earnings for the year:
 Add the regular annual earnings, the monetary reward, and the total raise.
 $\text{Total earnings} = 104000 + 5000 + 5200 = 114200$
 Therefore, the total amount of money John made that year is $\boxed{114200}$.

Case Study on GSM8K for Qwen2.5-Math-1.5B-Instruct. Steering one IPG SAE features with scaling factor $\gamma = 1.8$ on layer 20.

Question

Cars have lined up on the motorway. Some of the cars drive through in the first 15 minutes of the traffic jam, then 20 more cars drive through in the remaining 15 minutes of the jam. 5 cars from the line take an exit so they don't have to drive through the traffic jam. If there were originally 30 cars on the motorway, how many cars drove through the traffic jam in the first 15 minutes?

Response
Original:

To find the number of cars that drove through the traffic jam in the first 15 minutes, we need to first determine the total number of cars that drove through the traffic jam.
We know that 20 cars drove through in the remaining 15 minutes.
Let's denote the number of cars that drove through in the first 15 minutes as ' x '.
Since 5 cars took an exit, we need to subtract these from the total number of n . Since 5 cars took an exit, the total number of cars that drove through the traffic jam is $(x + 20) - 5$.
We are given that there were originally 30 cars on the motorway.
 $(x + 20) - 5 = 30$. Simplifying the equation, we get $x + 20 - 5 = 30$.
 $nx + 15 = 30$. Subtracting 15 from both sides, we get $x = 15$. So, the number of cars that drove through the traffic jam in the first 15 minutes is 15.

Enhanced:

To find the number of cars that drove through the traffic jam in the first 15 minutes, we need to first determine the total number of cars that drove through the traffic jam.
We know that 20 cars drove through in the remaining 15 minutes.
Let's denote the number of cars that drove through in the first 15 minutes as ' x '.
Since 5 cars took an exit, the total number of cars that drove through the traffic jam is the original number of cars minus the cars that took an exit, which is $30 - 5 = 25$.
We also know that the total number of cars that drove through the traffic jam is the sum of the cars that drove through in the first 15 minutes and the cars that drove through in the remaining 15 minutes.
To solve for ' x ', we subtract 20 from both sides of the equation: $x + 20 = 25$.
However, this is the number of cars that drove through the traffic jam in the first 15 minutes, but it does not account for the fact that the question asks for the number of cars that drove through the traffic jam in the first 15 minutes, not the number of cars that were in the first 15 minutes.
Since 5 cars took an exit, the number of cars that were in the first 15 minutes is $30 - 5 = 25$.
However, we know that 20 cars drove through in the remaining 15 minutes.
So, the number of cars that drove through in the first 15 minutes is $25 - 20 = 5$.

Case Study on GSM8K for Llama-3.1-8B-Instruct. Steering 5% IPG neurons with scaling factor $\gamma = 1.1$ on layer 22.

Question

A Senate committee has 5 Democrats, 5 Republicans, and 1 Independent. In how many ways can they sit around a circular table if all the members of each party all sit next to each other? (Two seatings are considered equivalent if one is a rotation of the other.)

Response
Original:

To solve this problem, we need to consider the restrictions and the total number of arrangements.
 1. **Total number of people**: There are 11 people in total (5 Democrats, 5 Republicans, and 1 Independent).
 2. **Restrictions**: All members of each party must sit next to each other. This means we can treat each party as a single unit when arranging the people around the table.
 3. **Circular table**: Since the table is circular, we need to account for the fact that rotations of the same arrangement are considered equivalent. This means we will have to divide the total number of arrangements by the number of people (11) to avoid overcounting.
 4. **Arranging the parties**: There are 3 parties (Democrats, Republicans, and Independent), which can be arranged in $3 = 6$ ways around the table.
 5. **Arranging the members within each party**: Within each party, the members can be arranged in the following ways:
 - Democrats: 5 ways
 - Republicans: 5 ways
 - Independent: 1 way (since there is only 1 member)
 6. **Total number of arrangements**: The total number of arrangements is the product of the number of ways to arrange the parties and the number of ways to arrange the members within each party:

$$\text{Total} = 6 * 5 * 5 * 1$$

 7. **Accounting for rotations**: Since the table is circular, we need to divide the total number of arrangements by the number of people (11) to avoid overcounting:

$$\text{Total} = (6 * 5 * 5 * 1) / 11$$

 8. **Calculating the total**: Now, let's calculate the total number of arrangements:

$$5 = 5 * 4 * 3 * 2 * 1 = 120$$

$$\text{Total} = (6 * 120 * 120 * 1) / 11$$

$$\text{Total} = (6 * 14400) / 11$$

$$\text{Total} = 86400 / 11$$

$$\text{Total} = 7872$$

 So, the total number of ways the Senate committee can sit around a circular table with all members of each party sitting next to each other is:

$$\boxed{7872}$$
.

Enhanced:

To solve this problem, we need to consider the restrictions and the total number of arrangements.
 1. The members of each party must sit together. This means we can treat each party as a single unit. So, we have 3 units: Democrats, Republicans, and the Independent.
 2. Since the table is circular, we need to account for the fact that rotations of the same arrangement are considered equivalent. This means we will have to divide the total number of arrangements by the number of people (or units) in each arrangement.
 3. First, let's calculate the number of ways to arrange these 3 units around the circular table. There are $(3-1) = 2 = 2$ ways to arrange the units.
 4. Now, let's consider the arrangements within each party. There are 5 ways to arrange the Democrats, 5 ways to arrange the Republicans, and 1 way to arrange the Independent (since there is only one Independent).
 5. To find the total number of arrangements, we multiply the number of ways to arrange the units by the number of ways to arrange the members within each party: $2 * 5 * 5 * 1$.
 Now, we calculate the total number of arrangements: $2 * 5 * 5 = 2 * 120 * 120 = 28800$.
 Therefore, the total number of ways the Senate committee can sit around a circular table if all the members of each party all sit next to each other is $\boxed{28800}$.

Models	Original	IPG (Enhanced)	ReFT
Qwen2.5-Math1.5B-Instruct	82.41	84.76	62.47
Llama3.1-8B-Instruct	85.89	87.41	77.94

Table 14. Performance comparison across models under different methods.

D.7. Comparison with Training-based Behavior Control for Long-Horizon Reasoning

Experimental Setup. To demonstrate our IPG are good at handling long-term dependencies, we additionally adopt ReFT (Wu et al., 2024) as a baseline. ReFT is a powerful training-based method for controlling model behavior by optimizing a language modeling loss on representation space. Specifically, to ensure a fair comparison, we trained ReFT on the same single layer (Layer 22 for Qwen model, Layer 24 for Llama model) using 1,000 samples from GSM8K train split. We used the best hyperparameter configuration for ReFT. We adopt the same evaluation settings in our paper (greedy decoding, zero-shot on GSM8K test set) and report only enhancement results as ReFT does not support suppression experiment.

Results and Analysis. As shown in Table 14, IPG consistently outperforms ReFT on both models under our setting. Notably, ReFT leads to a performance drop under our setting. We hypothesize three reasons for this degradation. (i) ReFT loss optimizes language modeling, not reasoning. (ii) Overfitting on training data. (iii) Data inefficiency of training-based method. Based on the empirical results, we confirm the advantages of IPG over handling long-term dependencies in multi-step reasoning scenario.

D.8. Behavior-control beyond Reasoning

D.8.1. REFLECTION BEHAVIOR

Experimental Setup. To demonstrate flexibility the signal function of J , we instantiate it to capture *reflection*. We utilize a set of typical keywords associated with reflective behavior serving as the signal (e.g., “wait”, “but”, “double check”). For the signal function, we calculate the total number of times these words appear in the trajectory τ , while preserving their token position information.

Results and Analysis. We conduct experiments using DeepSeek-Qwen-1.5B (Guo et al., 2025) on the AIME 2024 benchmark (Mathematical Association of America, 2024). As shown in Table 15, our intervention significantly increases the average number of reflection tokens per sample from 113.9 to 156.5. Consequently, this enhanced reflective behavior drives an improvement in final task accuracy, rising from 16.67% to 20.00%. This result confirms that our framework can effectively amplify specific mechanistic behaviors to improve complex reasoning outcomes.

Table 15. Impact of intervening on reflection-based signals on AIME 2024. We report the task accuracy and the average number of reflection tokens per sample (calculated over 30 samples).

Method	Accuracy (%) ↑	Reflection Tokens (Avg.) ↑
Original	16.67	113.9
Intervened (Ours)	20.00	156.5

D.8.2. SAFETY

Experimental Setup. Additionally, we try to attribute and control safety awareness. We instantiate the signal function J as a binary reward with regard to safety. For each prompt, we generate model responses both *before* and *after* intervention. Each response is then classified as *harmful* or *non-harmful* using the WILDGUARD classifier (Han et al., 2024). We report the *harmlessness rate*, defined as

$$\text{Harmlessness Rate} = \frac{\#\text{non-harmful responses}}{\#\text{total responses}}.$$

A higher harmlessness rate indicates safer model behavior. As a baseline, we adopt the activation-based method used in our main experiments, which selects neurons with the highest activation values. Following the main paper, we use greedy

decoding for all generations. The ENHANCE setting amplifies components associated with reduced harmfulness (higher is better), while SUPPRESS attenuates these components (lower is better).

Results and Analysis. Table 16 reports the results on Qwen2.5-1.5B-Instruct. IPG consistently outperforms the activation-based baseline under both enhancement and suppression settings. These results suggest that IPG is able to identify internal components that causally contribute to harmful behavior, enabling more effective and controllable safety steering.

Table 16. Comparison of IPG and activation-based methods for controlling harmfulness on Qwen2.5-1.5B-Instruct using the WILDGUARD benchmark (Han et al., 2024).

Method	Original (%)	Enhance ↑ (%)	Suppress ↓ (%)
Activation	89.52	89.64	89.05
IPG (Ours)	89.52	90.27	85.81