From Outcome to Representation: Tracing Reasoning Mechanisms through Integrated Policy Gradient

Changming Li
ShanghaiTech University

Kaixing Zhang
ShanghaiTech University

Yingdong Shi
ShanghaiTech University

Haoyun Xu
ShanghaiTech University

Zheng Zhang
ShanghaiTech University

Kaitao Song
Apple

Kan Ren*
ShanghaiTech University

Abstract

Large Language Models (LLMs) demonstrate remarkable reasoning capabilities, yet the internal mechanisms driving these multi-step processes remain opaque. Existing mechanistic interpretability approaches often rely on superficial text-pattern co-occurrences or capture only short-term effects, struggling to trace the long-horizon, outcome-driven influence of internal components such as neurons or sparse features in multi-step reasoning. In this paper, we introduce Integrated Policy Gradient (IPG), a novel, training-free framework that brings policy-based attribution into mechanistic interpretability for LLM reasoning. Grounded in *outcome-oriented* and *sequential-influence-aware* principles, it backpropagates outcome-level, non-differentiable behavioral signals (e.g., reasoning outcome correctness) through entire inference trajectories, utilizing path integration to isolate outcome-relevant mediators at component level for reliable attribution. This provides a linear approximation of Causal Mediation Analysis (CMA) in reasoning. Empirical evaluations demonstrate that IPG achieves precise localization and reveals mechanistic insights by identifying sparse, concentrated subsets of internal components associated with specific reasoning behaviors. Our method also enables modulation of reasoning capabilities, offering a powerful and reliable framework for understanding LLM reasoning. The project page is at `https://foundation-model-research.github.io/IPG`.

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, a central question remains unanswered: *which internal components are critical to multi-step reasoning, and how do their contributions across reasoning steps shape the final task outcome.* This lack of transparency poses challenges for truthfulness, safety, and controllability in reasoning systems [Wang et al., 2025b].

Mechanistic interpretability studies, which characterize internal computations responsible for model behaviors, suggest that the internal representation space of LLMs encodes reasoning-related con-

*Corresponding author: `renkan@shanghaitech.edu.cn`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

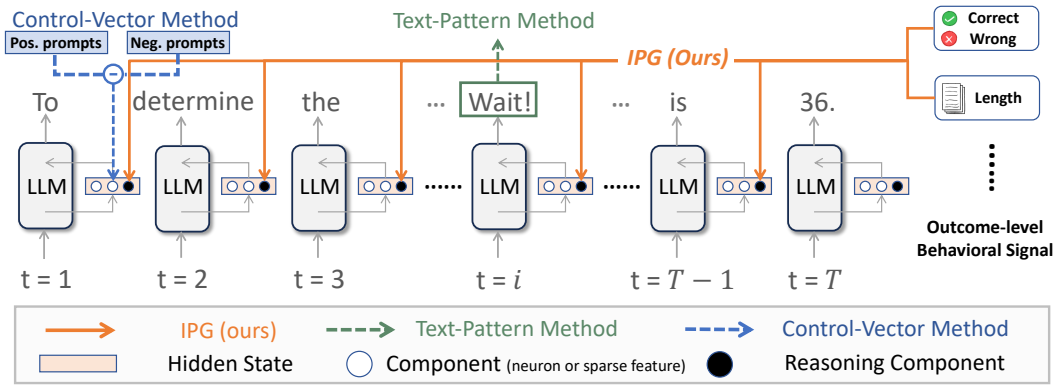


Figure 1: Interpretability paradigms for reasoning in LLMs, regarding (i) our proposed Integrated Policy Gradient method that attributes outcome-level behavioral signals. t is the reasoning time step; (ii) text-pattern methods that associate high activations with special tokens (e.g., "Wait"); and (iii) control-vector methods that rely on human-craft contrastive input pairs.

cepts [Højer et al., 2025, Galichin et al., 2025, Wang, 2026]. Existing interpretability approaches for LLM reasoning fall into three categories. The first analyzes individual internal components, such as neurons [Rai and Yao, 2024] or sparse 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 comprises conventional attribution methods, including Integrated Gradients (IG), NeuronSHAP, and Layer-wise Relevance Propagation (LRP) [Sundararajan et al., 2017, Ghorbani and Zou, 2020, Achtibat et al., 2024], which quantify the contributions of internal components to model predictions through gradient integration, Shapley-value estimation, or relevance propagation. The third derives control vectors to induce desired behaviors [Højer et al., 2025, Wang et al., 2025a], but they 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 model's inner components like neurons or sparse features [Huben et al., 2024] that genuinely drive reasoning behaviors. (ii) *Neglecting sequential influence*: they capture only short-term and local effects, e.g., arithmetic, instead of how sequential influence cumulatively contributes to the final outcome (e.g., task correctness).

To bridge this gap, we propose Integrated Policy Gradient (IPG), a novel training-free framework for attributing and interpreting reasoning in LLMs by identifying outcome-relevant components. IPG incorporates policy gradient [Mnih et al., 2016, Schulman et al., 2017] to back-propagate outcome-based behavioral signals through the entire inference trajectory to the model's internal components. This enables attribution of cumulative, long-range effects, overcoming challenges of sparse and non-differentiable outcome signals. IPG further accumulates policy gradients along the path from a baseline (e.g. zero value) to the original activation of each component for reliability in attribution [Sundararajan et al., 2017]. We compare our approach with existing interpretability paradigms in Figure 1. Additionally, Figure 2 highlights the specific reasoning components isolated by IPG.

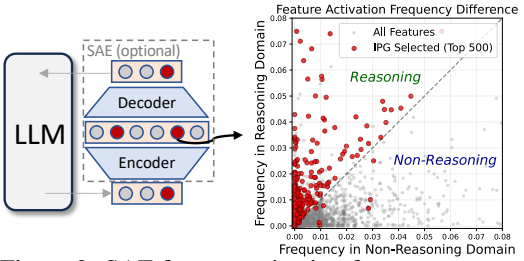


Figure 2: SAE feature activation frequency across reasoning and non-reasoning domains with IPG-selected features (red) and all features (gray).

From a causal perspective, IPG can be viewed as a tractable approximation to Causal Mediation Analysis (CMA) [Pearl, 2001], which quantifies the indirect effects of internal components (mediators) on model outcomes through counterfactual interventions. Directly estimating such effects requires repeated counterfactual evaluations, which becomes computationally prohibitive for long reasoning trajectories. By approximating these mediation effects without explicit counterfactual rollouts, IPG enables scalable component-level attribution of reasoning.

We evaluate and validate the components identified by IPG on representative open-source LLMs and its effectiveness in interpreting LLM reasoning. These components support improvements in reasoning performance and exhibit greater approximation on causal mediation than the competing attribution methods. Our effectiveness also transfers across datasets and domains. Beyond these evaluations, our spatial, semantic, and temporal analyses provide complementary views of the identified components: where they are concentrated, what reasoning-related patterns (e.g., logical planning or calculation) they are associated with, and when they become active during generation.

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. To the best of our knowledge, we are the first to bring policy-based attribution into mechanistic interpretability for LLM. (ii) We demonstrate that IPG provides mechanistic insights into how multi-step reasoning is internally organized. (iii) We show that IPG provides reliable and predictable modulation of key reasoning behaviors.

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 mainly include 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 these improvements, 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 and Gavves, 2024]. Reasoning, however, requires multi-step and long-horizon internal computation, posing unique challenges for the prevalent MI methods [Plaat et al., 2025]. Conventional attribution methods, including Integrated Gradients [Sundararajan et al., 2017], Neuron Shapley [Ghorbani and Zou, 2020], and LRP [Binder et al., 2016, Achtibat et al., 2024], identify components relevant to a single step, but do not explicitly address outcome-based credit assignment across generated trajectories. Reasoning-specific studies target reasoning-related concepts for internal components interpretation, including arithmetic [Rai and Yao, 2024] neurons and reasoning-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 careful annotations and prompt design, which struggle to ensure the intended target states are truly elicited [Højer et al., 2025]. In addition, [Stolfo et al., 2023] applies CMA to identify causal modules in reasoning, but they primarily focus on short-term tasks such as simple arithmetic. Overall, existing methods rely on co-occurrence correlations and fail 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

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 generates a reasoning trajectory $\tau = (a_1, a_2, \ldots, a_t, \ldots, a_T)$ 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 characterizes reasoning as a *multi-step* decision-making process with long-

term sequential dependency, where each token-generation time step can be associated with various reasoning behaviors.

Reasoning Component Space. LLM internal representations encode concepts related to reasoning behaviors [Galichin et al., 2025, Wang, 2026]. Our framework localizes these concepts at either the neuron-level (hidden states z_i) or the disentangled feature-level components (f_i) with Sparse Autoencoders (SAEs) [Huben et al., 2024].

Let $\mathbf{z}_t^l = [z_{t,1}^l, \ldots, z_{t,n}^l] \in \mathbb{R}^n$ denote the hidden state in layer l at generation step t under the input query $\mathbf{x}$, and $\mathbf{f}^l = [f_{t,1}^l, \ldots, f_{t,m}^l] \in \mathbb{R}^m$ its corresponding SAE feature representation (details in Appendix 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 $\mathbf{h}_t^l = [h_{t,1}^l, \ldots, h_{t,d}^l] \in \{\mathbf{z}_t^l, \mathbf{f}_t^l\}$, with $d = n$ for neuron-level and $d = m$ for feature-level.

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 simplicity, we omit the notation l to use $\mathbf{h}_t$ to represent target component in the subsequent part.

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

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).

Outcome-oriented 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 and Plank, 2024] and reasoning strength [Sheng et al., 2025]. Based on the outcome-oriented principle, we define a signal function $J(\cdot)$ to further help assess the model's internal components 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 Appendix D.10.

3.2 Integrated Policy Gradient (IPG) Method

Existing interpretability methods in LLM reasoning fall short due to three critical limitations: (i) Reliance on co-occurrences between activations and predefined text-patterns [Rai and Yao, 2024, Galichin et al., 2025] (ii) failure to capture *sequential-influence* characterized by cumulative outcomes [Højer et al., 2025]; (iii) exact CMA is intractable for long sequences, and standard gradient-based linear approximations [Mueller et al., 2026, Dai et al., 2022] struggle with sparse, non-differentiable feedback (e.g., final-answer correctness).

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

Policy Gradients in Representation Space. We 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. To obtain attribution of long-horizon 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 $\mathbf{h}_t$ (see Appendix A.4.1 for details). 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

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

where $s_t = (\mathbf{x}, a_1, \ldots, a_{t-1})$ represents the contextual state at time step t (consisting of the initial query $\mathbf{x}$ and all preceding prefix tokens) and a_t is the next token at the time step t. $\tau = (a_1, \ldots, a_T)$ 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 benefit of taking a_t under s_t. Empirically, this value is estimated based on the feedback derived from the signal function $J(\cdot)$.

Notably, 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.

IPG Attribution. Raw gradients can be noisy, capturing 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, it quantifies the mediation effect of an internal component on the final reasoning outcome. The integral accounts for the global contribution of a component by aggregating its effect across the entire interpolation path from a baseline to its original value.

Specifically, to quantify the influence of each component $h_{t,i}$ (the mediator) at step t on a reasoning-behavior measure J, we define the attribution score (details in Appendix A.4.2) as

$$\begin{aligned} \mathrm{IPG}(i; \mathbf{x}) &= (h_i - h_i') \int_0^1 \frac{\partial J}{\partial h_i(\alpha)} \, d\alpha \\ &= \sum_{t=1}^{T} (h_{t,i} - h_{t,i}') \int_0^1 \mathbb{E}_{\tau \sim \pi_\theta} \left[\frac{\partial}{\partial h_{t,i}} \log \pi_\theta\big(a_t \mid s_t;\, \mathbf{h}_t' + \alpha(\mathbf{h}_t - \mathbf{h}_t')\big) \cdot A^\pi(s_t, a_t) \right] d\alpha. \end{aligned} \tag{3}$$

where $\mathbf{h}_t'$ is the relative baseline value of $\mathbf{h}_t$ at step t, and $\mathbf{h}_t(\alpha) = \mathbf{h}_t' + \alpha(\mathbf{h}_t - \mathbf{h}_t')$ represents the intervened hidden state along the path. Deriving from Equation (2), for each generated trajectory $\tau = (a_1, \ldots, a_T)$, the log-probability of selecting token a_t under the interpolated state is weighted by the advantage function $A^\pi(s_t, a_t)$. This step allows us to backpropagate non-differentiable final reasoning outcomes (e.g., binary correctness) directly to intermediate internal components. We accumulate these policy gradients along the path from the baseline $\mathbf{h}'$ to the active state $\mathbf{h}$ [Sundararajan et al., 2017, Dhamdhere et al., 2019], capturing the global mediation effect rather than just local variance. IPG bypasses the limitations of exact CMA and standard linear approximations. It reduces gradient noise and produces robust signals for causal intervention. We provide ablation in Appendix D.4.

To ensure robustness, the IPG attribution score is computed per sample and aggregated across a dataset. Let $\mathcal{D} = \{\mathbf{x}^{(k)}\}_{k=1}^{M}$ be a small supporting dataset of M reasoning questions. For each component index i and each sample index k, we compute the per-sample attribution score $\mathrm{IPG}\big(i; \mathbf{x}^{(k)}\big)$ (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 $\mathbf{P} = \{i_1, \ldots, i_p\}$ are treated as the most influential components

$$S_i \;=\; \frac{1}{M} \sum_{k=1}^{M} \mathrm{IPG}\big(i; \mathbf{x}^{(k)}\big), \quad \mathbf{P} \;=\; \underset{i \in [d]}{\arg \text{top-p}}\; S_i \; . \tag{4}$$

Control Reasoning Behavior via IPG. We intervene on the components indexed by $i \in \mathbf{P}$ identified in Equation (4) in the model, following interpretability practices [Dai et al., 2022]. Formally, for each component $h_{t,i}$ at step t, we apply a multiplicative scaling factor as

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

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

Finally, at each step t, we substitute the original internal state $\mathbf{h_t}$ with the modified state vector $\tilde{\mathbf{h}}_\mathbf{t}$ 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 a new trajectory $\tilde{\tau}$ and response $\tilde{y}$:

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

Figure 2 visualizes this effect, and we empirically verify in Section 4.2 that the modified trajectory reflects the targeted modulation of reasoning behavior.

Discussion: Flexibility of IPG. Importantly, the IPG framework is *data-efficient*. The attribution process only needs to be performed *once* on a frozen model. After the set of components $\mathbf{P}$ is identified, interventions can be applied repeatedly during inference without any need for recomputing attributions. Furthermore, our empirical results confirm that IPG can accurately pinpoint the core components even with a relatively small dataset $\mathcal{D}$ (Appendix D.1.2). Besides, the mediator $\mathbf{h}$ is not restricted, but can represent any interpretable module within the network architecture. Finally, the framework can be seamlessly integrated with advanced policy gradient algorithms, such as GRPO [Shao et al., 2024], to further refine the estimation of reasoning outcomes (Appendix A.3).

4 Experiments

In this section, we empirically evaluate Integrated Policy Gradient (IPG) for attributing and interpreting reasoning behaviors in Large Language Models (LLMs). Our evaluation is driven by three core research questions (**RQs**) designed to progressively validate our approach. **RQ1 (Control Efficacy):** Can IPG identify internal components that exert a verifiable, sequential influence on reasoning behaviors? **RQ2 (Generalizability):** Are the identified reasoning components consistent and transferable across different benchmarks and domains? **RQ3 (Mechanistic Insights):** What deeper, fine-grained insights into the internal reasoning processes of LLMs does IPG reveal?

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]. 1

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 and 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.

Evaluation settings. Following prior work [Cheng et al., 2024, Tang et al., 2025], we consistently employ a deterministic *greedy* decoding strategy during inference to remove variance and ensure reproducible mechanistic analysis and also provide half-width 95% CI for IPG. To demonstrate that our method is also applicable under sampling setting, we also provide the sampling results in Appendix D.3. For all baselines, we apply at the same layer, search over the same number of parameters, and report the best-performing configuration for fair comparison.

Implementation Details. We constrain IPG and baseline methods on the residual stream following [Højer et al., 2025, Galichin et al., 2025] for fair comparison. More details are in Appendix B.

IPG settings. 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 reward model based signals (using Skywork-Reward-V2-Llama-3.1-8B [Liu et al., 2025]). For the baseline $\mathbf{h}'$ mentioned in Equation (3), we set $\mathbf{h}' = \mathbf{0} \in \mathbb{R}^n$, following [Dai et al., 2022]. We also provide analysis on choices of baseline in Appendix D.1.4.

4.2 Causal Intervention on Localized Components (RQ1)

We first validate whether the components localized by IPG genuinely drive reasoning behaviors, rather than only correlate with successful outputs.

Table 1: Ablation of IPG-identified components.

Task	Original	Random	IPG Feature
Reasoning	63.00	59.80 (-3.20)	28.80 (-34.2)
Non-Reasoning	12.11	9.81(-2.30)	10.81 (-1.3)

Finding 1: IPG identifies components with profound sequential influence. We examine whether IPG-identified components serve as a *outcome-aware mediator* in reasoning by (i) ablating them (setting their activation to 0) to test whether reasoning degrades, and (ii) amplifying them to test whether reasoning improves. We

Table 2: Accuracy (%) after enhancing components selected by IPG and baseline methods across reasoning benchmarks under greedy decoding. Results are reported for Qwen2.5-Math-1.5B-Instruct and Llama3.1-8B-Instruct. Values following ± denote the half-width of the 95% confidence interval, in percentage points. The best and second-best point estimates in each column are shown in **bold** and underlined, respectively; ties receive the same formatting.

Method	Qwen2.5-Math-1.5B				Llama3.1-8B			
	GSM8K	MATH	AIME	GPQA	GSM8K	MATH	AIME	GPQA
Original	82.41±0.65	63.00±0.95	10.00±3.17	24.75±1.92	85.89±0.72	41.80±0.29	6.67±3.17	23.23±0.96
Random	82.86±0.65	63.20±0.88	16.67±4.75	26.77±1.68	86.20±0.65	41.40±0.29	6.67±4.75	27.78±0.96
Activation	82.56±0.65	63.40±1.33	**20.00**±4.75	25.75±1.92	85.75±0.61	43.00±0.19	13.33±4.75	25.25±0.72
R.N. [Rai and Yao, 2024]	82.26±0.83	62.20±1.14	13.33±4.62	24.24±1.92	86.05±0.65	40.80±1.23	6.67±4.75	21.21±2.16
C.V. [Højer et al., 2025]	83.85±0.07	62.60±1.04	16.67±3.17	28.78±1.44	85.97±0.86	40.00±0.57	6.67±4.75	28.78±0.48
SAE-R [Galichin et al., 2025]	83.32±0.40	64.00±0.76	13.33±4.75	25.76±1.68	86.28±0.54	41.80±0.85	10.00±3.17	26.77±0.96
IPG (Ours)	84.61±0.40	65.80±0.88	**20.00**±3.17	33.33±1.68	87.18±0.43	42.40±0.66	13.33±4.62	27.27±0.96
IPG-SAE (Ours)	**84.76**±0.50	**66.00**±0.76	**20.00**±4.75	**35.35**±1.20	**87.41**±0.43	**44.00**±0.57	**20.00**±4.62	**29.80**±0.72

ablate the top-ranked IPG-identified features in Table 1. It shows a sharp drop on reasoning task (MATH-500 [Hendrycks et al., 2021]), with minor change on non-reasoning task (AlpacaEval [Dubois et al., 2024]), indicating that the localized components are genuinely critical for reasoning.

We further evaluate IPG alongside baseline methods across multiple reasoning datasets, applying enhancement interventions. In Table 2, IPG achieves consistent control over reasoning performance. Enhancing IPG-identified components yields consistent gains, while integrating IPG with Sparse Autoencoders (IPG-SAE) pushes effectiveness even further. Baselines relying on text patterns (R.N., SAE-R) or contrastive pairs (C.V.) yield marginal or unstable improvements, suggesting they only capture short term effect rather than sequential influence required by multi-step reasoning.

We additionally compare IPG with three representative component-attribution methods, including Integrated Gradients [Sundararajan et al., 2017], Neuron Shapley [Ghorbani and Zou, 2020], and AttnLRP [Achtibat et al., 2024], in the neuron intervention setting on Qwen2.5-Math-1.5B-Instruct. All three baselines improve accuracy over the original model, while IPG achieves the highest accuracy across the four evaluated benchmarks. These results extend the comparison beyond reasoning-specific baselines and support the utility of outcome-based sequential attribution. The full results and attribution settings are provided in Appendix D.5.

Finding 2: IPG alignments with a direct causal mediation proxy. To complement the enhancement experiments, we measured the alignment (overlap ratio) between the top neurons identified by our method with a direct mediation proxy (details in Appendix C.1). Let T^k_{CMA} denote the reference set of k neuron indices obtained from this proxy, and let T^k_m denote the top-k neuron indices selected by attribution method m. We measure their alignment as $\mathrm{Overlap@}\,k(m) = \frac{|T^k_{\mathrm{CMA}} \cap T^k_m|}{k}$.

As shown in Table 3, IPG achieves a substantially higher mean overlap than the attribution baselines on both models. These results suggest that IPG provides an approximation of causal mediation.

Table 3: Alignment with a group-based zero-ablation mediation proxy on a subset of MATH-500. Values are mean $\mathrm{Overlap@}\,k\pm$ standard deviation with k corresponding to approximately 5% of neurons. Qwen and DeepSeek denote Qwen2.5-Math-1.5B-Instruct and DeepSeek-R1-Distill-Qwen-1.5B, respectively.

Method	Qwen	DeepSeek
Random	0.064 ± 0.023	0.064 ± 0.023
IG [Sundararajan et al., 2017]	0.091 ± 0.025	0.066 ± 0.022
Neuron SHAP [Ghorbani and Zou, 2020]	0.072 ± 0.020	0.065 ± 0.005
AttnLRP [Achtibat et al., 2024]	0.082 ± 0.018	0.062 ± 0.013
IPG (Ours)	$\mathbf{0.305} \pm 0.113$	$\mathbf{0.246} \pm 0.030$

Finding 3: IPG refines the entire reasoning trajectory. To understand how interventions alter the generation process, we qualitatively compare responses from Qwen2.5-Math-1.5B-Instruct [Yang et al., 2024] on GSM8K before and after intervening on IPG-identified components. As illustrated in Figure 3, the baseline model fails to follow strict logical constraints. Amplifying IPG components does not merely force the correct final answer; instead, it fundamentally corrects the intermediate reasoning steps, resulting in a more coherent and rigorous logical trajectory.

Finding 4: IPG enables predictable and granular modulation. We evaluate the precise controllability of two facets of reasoning: *capability* (performance sensitivity) and *strength* (reasoning chain length). By varying the intervention scaling factor γ, IPG yields smooth, predictable control over reasoning capability, whereas baseline methods exhibit unstable dynamics (Figure 4a). Furthermore, for reasoning strength, IPG successfully modulates the length of the reasoning trajectory without

sacrificing task accuracy (Figure 4b), suggesting that IPG isolates the distinct internal components governing different dimensions of reasoning behavior.

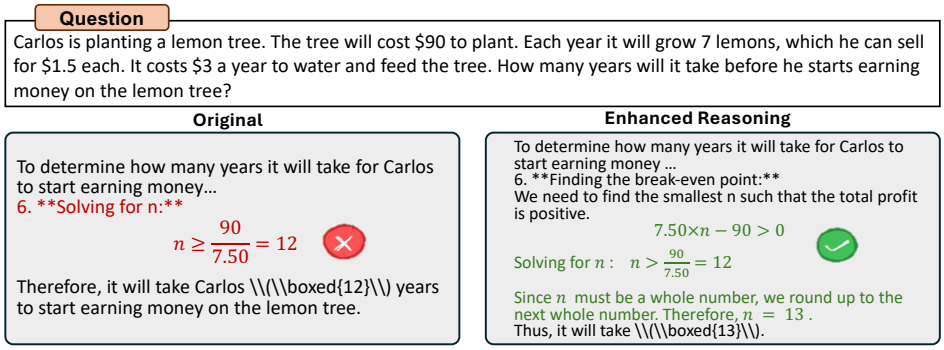
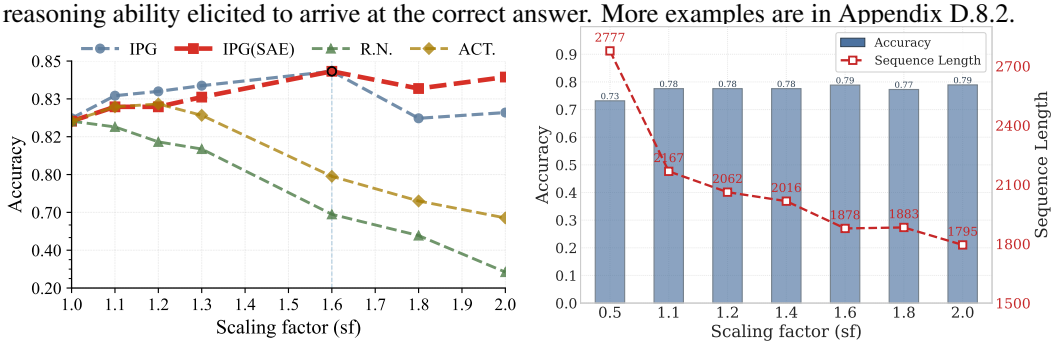


Figure 3: Responses generated by Qwen2.5-Math-1.5B-Instruct [Yang et al., 2024] on one example in GSM8K [Cobbe et al., 2021], including both original and intervened outputs, with the model's reasoning ability elicited to arrive at the correct answer. More examples are in Appendix D.8.2.



(a) Scaling factor effects.

(b) Reasoning strength effects.

Figure 4: 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: Reasoning strength control in GSM8K [Cobbe et al., 2021] for DeepSeek-R1-Distilled-Qwen-1.5B [Guo et al., 2025].

4.3 Generalizability of Reasoning Mechanisms (RQ2)

A key criterion for mechanistic discovery is *generalization*: components that genuinely implement reasoning should not be tied to a single dataset and domain. We therefore examine whether IPG-identified components transfer beyond the source task on which they are localized. Specifically, we evaluate both *cross-dataset transfer* within mathematical tasks and *cross-domain transfer* across mathematical, coding, and commonsense reasoning tasks.

Finding 5: IPG identifies reasoning mechanisms with cross-dataset and cross-domain consistency. Components localized on MATH500 [Hendrycks et al., 2021] transfer (directly intervene) effectively to GSM8K [Cobbe et al., 2021], outperforming baselines in cross-dataset generalization. Remarkably, this efficacy extends across distinct domains, including coding (HumanEval) and commonsense reasoning (CSQA) (Table 4). Supported by Dice similarity metrics (Figure 8, Appendix D.6), these results suggest that IPG captures core and transferable reasoning ability.

4.4 Deeper Mechanistic Insights (RQ3)

Having established the efficacy and generalizability of the identified components, we leverage IPG to unpack the mechanisms of LLM reasoning at a granular level. Specifically, we investigate where these components are located (Section 4.4.1), what reasoning functions they encode (Section 4.4.2), and when they contribute to the reasoning process (Section 4.4.3).

4.4.1 Spatial Localization and Concentration

Finding 6: Reasoning-related components are spatially localized and concentrated. We first examine *where* influential components are located. As shown in Figure 5a, performance gains vary

Table 4: Cross-dataset (mathematics) and cross-domain (code and commonsense) transfer under greedy decoding. Best and second best results are shown in **bold** and underlined format. We provide the half-width 95% CI for our method.

	Cross-Dataset		Cross-Domain			
	MATH-500 → GSM8K		MATH-500 → HumanEval		HumanEval → CSQA	
Method	**Acc. (%) ↑**	**Avg. Tok.**	**Acc. (%) ↑**	**Avg. Tok.**	**Acc. (%) ↑**	**Avg. Tok.**
Original	82.41	319.68	11.58	212.12	21.14	317.05
Activation	82.94	317.17	12.80	161.93	20.00	276.58
R.N. [Rai and Yao, 2024]	82.87	312.63	15.24	163.98	20.26	258.78
C.V. [Højer et al., 2025]	83.39	320.03	14.02	195.33	20.96	275.47
IPG (Ours)	**84.08**±0.32	315.86	**21.34**±1.45	191.40	**22.72**±0.54	325.89

across layers and intervention widths, with the largest gains concentrated in specific middle-to-late layers. This pattern suggests that components with substantial effects on reasoning performance are not uniformly distributed throughout the model.

The cumulative attribution scores in Figure 5b, computed by summing S_i over selected components $\mathbf{P}$ in Equation (4), further show that a relatively small subset of components accounts for a large share of attribution importance within these layers. Together, the attribution concentration and intervention gains identify localized regions whose components are particularly influential for reasoning.

4.4.2 Semantic Analysis

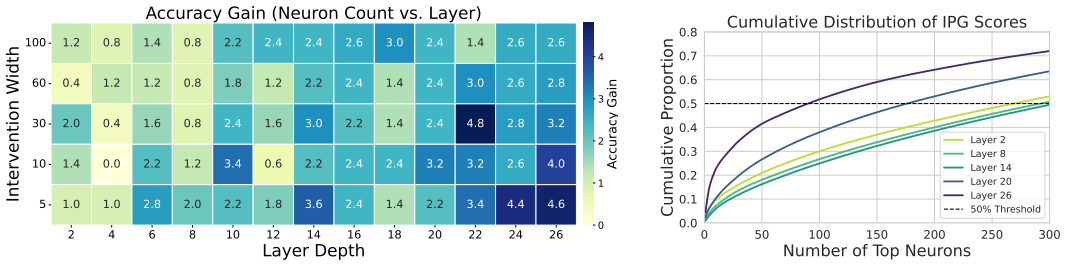
Finding 7: IPG-selected features exhibit distinct reasoning-related semantic patterns. Having identified where influential components are concentrated, we next examine *what* they represent. Individual neurons can entangle multiple concepts [Mu and Andreas, 2020, Olah et al., 2020], so we analyze IPG-selected features in the Sparse Autoencoder (SAE) representation space. For each feature, we collect its highest-activation contexts across GSM8K.

As shown in Table 5, these contexts reveal distinct reasoning-related patterns. Feature #7349 activates during explicit arithmetic calculations, whereas Features #486 and #12179 appear in planning and summarization contexts, respectively. Additional examples are provided in Appendix D.8.1. These differences suggest that the selected features are associated with complementary aspects of problem solving, providing interpretable descriptions of the selected components.

4.4.3 Temporal Organization

Finding 8: IPG-selected features exhibit distinct temporal activation patterns. Finally, we examine *when* these features are active. We track representative SAE features across reasoning trajectories, grouping their activations into step deciles of normalized generation progress. As shown in Figure 6, different features exhibit distinct activation profiles, with activity concentrated at different stages rather than uniformly distributed throughout generation.

These temporal differences complement the semantic patterns in Table 5: the selected features differ not only in their activating contexts, but also in when they are engaged during a generated solution (e.g., planning or calculating). Overall, the spatial, semantic, and temporal analyses characterize



(a) Performance Gain by Depth and Width

(b) IPG Score Cumulation

Figure 5: Layer-wise localization and concentration of reasoning components. **(a)** The heatmap for reasoning task (MATH-500) performance enhancement across different network layers and intervention widths (number of targeted neurons). **(b)** The cumulative IPG scores for selected layers.

Feat. ID	Peak Context
486	Let's start with the **first step**:
9889	Therefore, **the answer** is $\boxed{14}$.
7349	25% of 80 is **0.25** * **80** = 20,
12179	**Therefore**, the total amount of money Maddy will spend...
11155	The **total number** of rows of flowers Jackson will plant is...

Table 5: SAE features identified by IPG alongside the text contexts. Tokens receiving the maximal activation value are in **bold**.

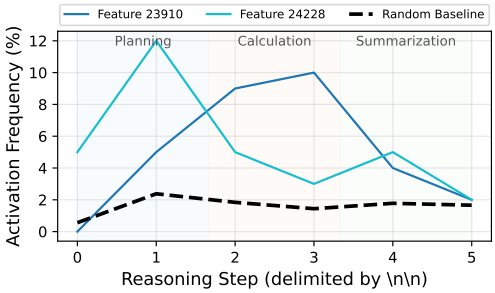


Figure 6: Temporal activation dynamics of SAE features, measured across normalized reasoning progress discretized into step deciles.

Table 6: Enhancing neurons from prompt-elicited to reason-distilled models without re-running IPG.

Model	GSM8K	MATH500	AIME2024	AIME2025
DeepSeek-R1-Distill-Qwen-1.5B	75.21	60.60	16.67	13.33
+ IPG Neuron Transferring	**78.01**	**63.80**	**30.00**	**30.00**

where influential components are concentrated, what patterns they are associated with, and when they become active.

4.4.4 Mechanisms Under Distillation

Finding 9: Distillation preserves and amplifies base mechanisms. Finally, we transfer IPG-identified neurons from a standard prompt-elicited model (Qwen2.5-Math-1.5B-Instruct) directly to its reasoning-distilled counterpart (DeepSeek-R1-Distilled-Qwen-1.5B), **without** re-running IPG. As shown in Table 6, enhancing the inherited neurons within the DeepSeek-distilled model consistently improves reasoning performance. This insight suggests that modern model distillation may preserve the internal architecture for reasoning pathways and it relies on and strengthens the fundamental reasoning mechanisms in the base model that are already localized by IPG.

5 Conclusion

In this work, we introduced Integrated Policy Gradient (IPG), a novel, training-free framework for interpreting LLM reasoning guided by *outcome-oriented* and *sequential-influence-aware* principles. By bridging Causal Mediation Analysis with policy gradients [Schulman et al., 2017] and path integration, IPG successfully back-propagates sparse, non-differentiable outcome signals (e.g., final task correctness) to isolate outcome-relevant mediators across long-horizon trajectories. Empirically, our evaluations revealed that complex reasoning relies on concentrated regions and exhibits specialized behaviors. Beyond enabling predictable and granular control over reasoning behavior, IPG demonstrated robust cross-dataset and cross-domain transferability. Future work can explore extending this framework to other domains such as emotional intelligence and creativity and further refining the manipulation of these localized components.

Acknowledgments and Disclosure of Funding

The research was supported by the National Natural Science Foundation of China (Grant No. 62406193) and the ShanghaiTech AI Initiative (Grant No. AI2026B08). The authors gratefully acknowledge further assistance from the Key Laboratory of Intelligent Perception and Human-Machine Collaboration (Shanghaitech University), Ministry of Education, and the HPC Platform of ShanghaiTech University. This research was also supported by Meituan.

There are no competing interests to declare.

References

Reduan Achtibat, Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Aakriti Jain, Thomas Wiegand, Sebastian Lapuschkin, and Wojciech Samek. AttnLRP: Attention-aware layer-wise relevance propagation for transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 135–168. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/achtibat24a.html.

Max Belitsky, Dawid J. Kopiczko, Michael Dorkenwald, M. Jehanzeb Mirza, Cees G. M. Snoek, and Yuki M. Asano. Kv cache steering for inducing reasoning in small language models. *arXiv preprint arXiv:2507.08799*, 2025.

Leonard Bereska and Stratis Gavves. 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.

Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In Alessandro E.P. Villa, Paolo Masulli, and Antonio Javier Pons Rivero, editors, *Artificial Neural Networks and Machine Learning – ICANN 2016*, pages 63–71, Cham, 2016. Springer International Publishing. ISBN 978-3-319-44781-0.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 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.

Xiaoxue Cheng, Junyi Li, Wayne Xin Zhao, Hongzhi Zhang, Fuzheng Zhang, Di Zhang, Kun Gai, and Ji-Rong Wen. Small agent can also rock! empowering small language models as hallucination detector, 2024. URL https://arxiv.org/abs/2406.11277.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. Knowledge neurons in pretrained transformers. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 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/.

Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. How important is a neuron. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SylKoo0cKm.

Yann Dubois, Percy Liang, and Tatsunori Hashimoto. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=CybBmzWBX0.

Subhabrata Dutta, Joykirat Singh, Soumen Chakrabarti, and Tanmoy Chakraborty. 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=uHLDkQVtyC.

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

Andrey Galichin, Alexey Dontsov, Polina Druzhinina, Anton Razzhigaev, Oleg Y. Rogov, Elena Tutubalina, and Ivan Oseledets. 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.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Leo Gao, Tom Dupre la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tcsZt9ZNKD.

Amirata Ghorbani and James Zou. Neuron shapley: discovering the responsible neurons. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. Wildguard: Open one-stop moderation tools for safety risks, jailbreaks, and refusals of llms. *Advances in Neural Information Processing Systems*, 37:8093–8131, 2024.

Zhenghao He, Guangzhi Xiong, Bohan Liu, Sanchit Sinha, and Aidong Zhang. Reasoning beyond chain-of-thought: A latent computational mode in large language models. *arXiv preprint arXiv:2601.08058*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

Bertram Højer, Oliver Simon Jarvis, and Stefan Heinrich. Improving reasoning performance in large language models via representation engineering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IssPhpUsKt.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 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.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed prompting: A modular approach for solving complex tasks, 2023. URL https://arxiv.org/abs/2210.02406.

Jack Lindsey, Wes Gurnee, Emmanuel Ameisen, Brian Chen, Adam Pearce, Nicholas L. Turner, Craig Citro, David Abrahams, Shan Carter, Basil Hosmer, Jonathan Marcus, Michael Sklar, Adly Templeton, Trenton Bricken, Callum McDougall, Hoagy Cunningham, Thomas Henighan, Adam Jermyn, Andy Jones, Andrew Persic, Zhenyi Qi, T. Ben Thompson, Sam Zimmerman, Kelley Rivoire, Thomas Conerly, Chris Olah, and Joshua Batson. On the biology of a large language model. Transformer Circuits Blog, March 27 2025. Anthropic; https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-reward-v2: Scaling preference data curation via human-ai synergy, 2025. URL https://arxiv.org/abs/2507.01352.

George Ma, Zhongyuan Liang, Irene Y Chen, and Somayeh Sojoudi. Do sparse autoencoders identify reasoning features in language models? *arXiv preprint arXiv:2601.05679*, 2026.

Alireza Makhzani and Brendan Frey. K-sparse autoencoders. In *International Conference on Learning Representations*, 2014.

Samuel Marks, Can Rager, Eric J Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 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.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

Philipp Mondorf and Barbara Plank. 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=Lmjgl2n11u.

Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *CoRR*, abs/2006.14032, 2020. URL https://arxiv.org/abs/2006.14032.

Aaron Mueller, Jannik Brinkmann, Millicent L. Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. 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.

Aaron Mueller, Jannik Brinkmann, Millicent Li, Samuel Marks, Koyena Pal, Nikhil Prakash, Can Rager, Aruna Sankaranarayanan, Arnab Sen Sharma, Jiuding Sun, Eric Todd, David Bau, and Yonatan Belinkov. The quest for the right mediator: Surveying mechanistic interpretability for nlp through the lens of causal mediation analysis. *Computational Linguistics*, 52(1):331–378, 03 2026. ISSN 0891-2017. doi: 10.1162/COLI.a.572. URL https://doi.org/10.1162/COLI.a.572.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 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.

Judea Pearl. Direct and indirect effects. In *Proceedings of the 17th conference on Uncertainty in artificial intelligence*, pages 411–420. Morgan Kaufmann Publishers Inc., 2001.

Aske Plaat, Annie Wong, Suzan Verberne, Joost Broekens, Niki Van Stein, and Thomas Back. 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.

Daking Rai and Ziyu Yao. An investigation of neuron activation as a unified lens to explain chain-of-thought eliciting arithmetic reasoning of LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 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/.

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

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Leheng Sheng, An Zhang, Zijian Wu, Weixiang Zhao, Changshuo Shen, Yi Zhang, Xiang Wang, and Tat-Seng Chua. 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=H26A7cl91u.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 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.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328, 2017.

Xinyu Tang, Xiaolei Wang, Zhihao Lv, Yingqian Min, Wayne Xin Zhao, Binbin Hu, Ziqi Liu, and Zhiqiang Zhang. Unlocking general long chain-of-thought reasoning capabilities of large language models via representation engineering, 2025. URL https://arxiv.org/abs/2503.11314.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Mengru Wang, Ziwen Xu, Shengyu Mao, Shumin Deng, Zhaopeng Tu, Huajun Chen, and Ningyu Zhang. Beyond prompt engineering: Robust behavior control in LLMs via steering target atoms. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 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/.

Wenshuo Wang. Llm reasoning is latent, not the chain of thought. *arXiv preprint arXiv:2604.15726*, 2026.

Yanbo Wang, Yongcan Yu, Jian Liang, and Ran He. A comprehensive survey on trustworthiness in reasoning with large language models. *arXiv preprint arXiv:2509.03871*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 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.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning, 2023. URL https://arxiv.org/abs/2309.05653.

Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. 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).

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:

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

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 $\mathbf{P}$

```
for j = 1 to M do
    Extract hidden states z = [z_1, z_2, ..., z_n] from F(x_j)
    Compute components h = [h_1, h_2, ..., h_d] = Φ(z)
    for i = 1 to m do
        S(h_i; X) ← S(h_i; X) + IPG(i; x_j) {Equation (3)}
    end for
end for
P ← arg top-p_{i∈[m]} S(u_i; X) {Equation (4)}
return 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 $\mathbf{s}$ 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 $\mathbf{s}'$ 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 support different algorithms of policy gradient [Mnih et al., 2016], as mentioned in Section 3.2.

Algorithm 2 IPG: Controlling Reasoning Behavior over Generation Steps

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

for $t = 1$ to T **do**
  Compute components $\mathbf{h}_t = [h_{t,1}, \ldots, h_{t,d}] = \Phi(\mathbf{z}_t)$ {Equation (23)}
  Initialize $\mathbf{h}'_t \leftarrow \mathbf{h}_t$
  for $k = 1$ to p **do**
    $h'_{t,i_k} \leftarrow \gamma \cdot h'_{t,i_k}$ {Scaling, Equation (5)}
  end for
  if Φ is Identity **then**
    $\mathbf{z}_t \leftarrow \mathbf{h}'_t$
  else if Φ is SAE-encoder **then**
    $\boldsymbol{\epsilon}_t \leftarrow \mathbf{z}_t - (\mathbf{W}_{\text{dec}}\mathbf{h}_t + \mathbf{b}_{\text{dec}})$
    $\mathbf{z}_t \leftarrow \mathbf{W}_{\text{dec}}\mathbf{h}'_t + \mathbf{b}_{\text{dec}} + \boldsymbol{\epsilon}_t$ {Equation (23)}
  end if
end for
Return $\{\mathbf{z}_t\}_{t=1}^T$

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

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

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].

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

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:

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

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

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

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

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

where $\tau = (s_1, a_1, \ldots, s_T, a_T)$ 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:

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

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

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

we obtain

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

Since the trajectory probability factorizes as

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

the gradient can be written as

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

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

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

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

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

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 $\pi_\theta(\tau \mid \mathbf{x})$. 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 $\alpha \in [0, 1]$ [Sundararajan et al., 2017, Dhamdhere et al., 2019]. The choices of baseline h' are usually counterfactual values including $\mathbb{E}_{x\in\mathcal{D}}[h_i]$ or just zero value, serving as mean or zero ablation [Mueller et al., 2026]. We also provide our intervention results in Appendix D.1.4, which suggests our robustness under different choices of baseline value.

We define the objective function $\mathcal{J}(\alpha)$ as the *expected reward* conditioned on this interpolated state: $\mathcal{J}(\alpha) := \mathbb{E}_{\tau\sim\pi_\theta(\cdot|x;h(\alpha))}[J(\tau)]$, where $J(\tau)$ is the non-differentiable metric (e.g., correctness). To attribute the change in this objective, we first derive its gradient as $\nabla_{h(\alpha)}\mathcal{J}(\alpha) =$

$\mathbb{E}_\tau[\sum_t \nabla_{h(\alpha)} \log \pi_\theta(a_t \mid s_t; h(\alpha)) A_t^\pi]$. Since $h(\alpha) = h' + \alpha(h - h')$, we have $\frac{dh}{d\alpha} = h - h'$. By the chain rule,

$$\frac{d\mathcal{J}(\alpha)}{d\alpha} = \left\langle \nabla_{h(\alpha)} \mathcal{J}(\alpha),\ \frac{dh(\alpha)}{d\alpha} \right\rangle = \left\langle \nabla_{h(\alpha)} \mathcal{J}(\alpha),\ h - h' \right\rangle. \tag{20}$$

This is a scalar function of α. The inner product is a sum over the representation dimensions $i = 1, \ldots, d$,

$$\left\langle \nabla_{h(\alpha)} \log \pi_\theta(\tau \mid h(\alpha)),\ h - h' \right\rangle = \sum_{i=1}^{d} (h_i - h'_i) \nabla_{h_i(\alpha)} \log \pi_\theta(\tau \mid h(\alpha)), \tag{21}$$

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:

$$\begin{aligned} \mathcal{J}(1) - \mathcal{J}(0) &= \int_0^1 \frac{d\mathcal{J}(\alpha)}{d\alpha}\, d\alpha = \int_0^1 \langle \nabla_{h(\alpha)} \mathcal{J}(\alpha), h - h' \rangle\, d\alpha \\ &= \sum_{i=1}^{d} (h_i - h'_i) \int_0^1 \mathbb{E}_{\tau \sim \pi_\theta(\cdot \mid h(\alpha))} \left[\nabla_{h_i(\alpha)} \log \pi_\theta(\tau \mid h(\alpha)) A^\pi(\tau) \right] d\alpha. \end{aligned} \tag{22}$$

This derivation confirms that our attribution score IPG_i represents the functional contribution of the i-th component to the expected return, satisfying $\sum_i \text{IPG}_i = \mathbb{E}[J]_{\text{original}} - \mathbb{E}[J]_{\text{baseline}}$.

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 and 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

$$\mathbf{f} = [f_1, f_2, \ldots, f_m] = \text{TopK}(\mathbf{W}_{\text{enc}}(\mathbf{z} - \mathbf{b}_{\text{pre}}))\ , \quad \hat{\mathbf{z}} = \mathbf{W}_{\text{dec}}\mathbf{f} + \mathbf{b}_{\text{pre}} + \epsilon(\mathbf{z}), \tag{23}$$

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 (23), the k-SAE contains an encoder $\mathbf{W}_{\text{enc}}$ and a decoder $\mathbf{W}_{\text{dec}}$ initialized with the same parameters. The reconstruction loss we use is a standard mean squared error (MSE) loss defined as

$$\mathcal{L}(\mathbf{z}) = \|\mathbf{z} - \hat{\mathbf{z}}\|_2^2 \tag{24}$$

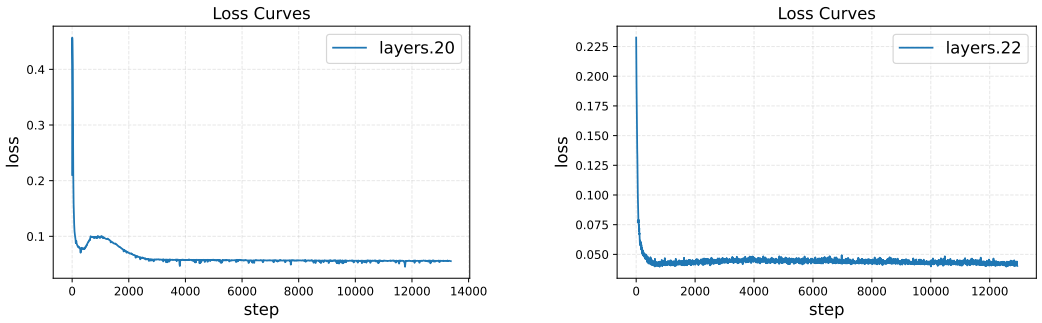
where $\mathbf{z}$ is the original hidden state vector and $\hat{\mathbf{z}}$ denotes its reconstruction (excluding the residual error term). The dimension of the sparse representation space is 24576 for Qwen2.5-Math-1.5B-Instruct [Yang et al., 2024] and 65536 for Llama-3.1-8B-Instruct [Grattafiori et 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 and 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

$$FVU = \frac{\mathcal{L}(\mathbf{z})}{\text{var}[\mathbf{z}]} \tag{25}$$

where $\mathbf{z}$ is the hidden state, $\mathcal{L}(\mathbf{z})$ is defined in Equation (24) 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 performs well on the FVU metric.



(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

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 hyperparameter used for the attribution method throughout our experiments. Unless otherwise specified, the same configuration is applied across all datasets and models. The attribution is on one single NVIDIA A40 GPU for 2 hours for Qwen2.5-Math-1.5B-Instruct [Yang et al., 2024] and one single NVIDIA H20 GPU for Llama3.1-8B-Instruct [Grattafiori et al., 2024]

Table 7: IPG-GRPO Hyperparameter Settings.

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

B.3 Fine-grained Reasoning Mechanism Discovery

B.3.1 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

$$\text{Freq}(f) \;=\; \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \mathbb{I}(f_t > 0). \tag{26}$$

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.

For CMA proxy, we repeat the evaluation with five random seeds on both Qwen2.5-Math-1.5B-Instruct and DeepSeek-R1-Distill-Qwen-1.5B. The latter provides a longer-trajectory setting, with approximately 4,000 generated tokens on average on MATH-500, compared with approximately 550 for Qwen. We construct a group-based mediation proxy by randomly sampling neuron groups, zero-ablating their activations at a target layer during generation, and selecting groups with high drop of reasoning ability measured by externally verified correctness.

C.2 Baselines

For the baseline, all provided results are fair, including searching for their best configurations according to their presented hyperparameter selection rules.

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 with the same number of our IPG. 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 and 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 employs Sparse Autoencoders (SAEs) [Makhzani and 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 experiments and studies for better illustration of our IPG.

D.1 Sensitivity Analysis

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 8. The results are based on Qwen2.5-Math-1.5B-Instruct [Yang et al., 2024] layer 20.

Table 8: 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 3.2.

k	**GSM8K Acc. (Enhance)** $\uparrow$
Original	82.41
8	84.38
32	**84.76**
64	83.70

In Table 8, 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 9, 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.

Table 9: Evaluation on the impact of varying the top-p number of neurons in 3.2 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 $k = 32$.

p	**GSM8K Acc. (Enhance)** $\uparrow$
Original	82.41
10	82.94
20	83.85
30	83.78
40	83.85
50	**84.60**
60	84.15

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 10): 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

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 3.2) is applied at the 24th layer. As shown in Table 11, reward model-based signals outperform rule-based

Table 10: 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%.

# Samples	**GSM8K Acc. (Enhance)** $\uparrow$
1	82.79
10	83.17
20	83.17
50	83.54
100	83.39
500	**83.55**

Table 11: Performance of IPG in enhancing accuracy across reasoning datasets for Qwen2.5-Math-1.5B-Instruct [Yang et al., 2024] under different reward signals.

Model	Reward Method	GSM8K	MATH-500	AIME-2024	GPQA Diamond
Qwen2.5-Math-1.5B	Rule Function	84.15	64.20	20.00	26.77
	Reward Model	84.60	65.80	20.00	35.35

Table 12: 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%.

$\mathbf{h}'$	GSM8K Acc. (Enhance) ↑
Zero vector	**84.60**
Mean hidden state	84.45
Random noise	84.00

signals in most datasets, suggesting that reward models capture richer information or better represent reasoning outcomes, thereby enhancing IPG's effectiveness.

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 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 12. 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

$$J(\tau) = f(|\tau|) \,, \tag{27}$$

where $f(\cdot)$ can be any monotonically increasing function. For example, we can have a quadratic 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 4b 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.

Table 13: Comparison of IPG and baseline methods on pass@8 accuracy across reasoning datasets for Qwen2.5-Math-1.5B-Instruct [Yang et al., 2024]

Method	GSM8K	MATH-500	AIME-2024
Original	93.83	76.38	25.07
IPG (Ours)	**94.77**	**77.77**	**32.61**

D.3 Validation of Reasoning Components under sampling evaluation.

As discussed in Section 4.1, our method remains effective under sampling-based evaluation, where multiple (16) candidate solutions are generated. Table 13 reports the accuracy on GSM8K, MATH-500, and AIME-2024 under such a setting. We compare the original model (without our reasoning-strength control) against our method using the same sampling budget. Across all three benchmarks, our method yields consistent improvements. These results confirm that the reasoning components governed by our approach are beneficial not only in greedy decoding but also when exploration is encouraged through sampling.

D.4 Ablation on Policy Gradient

In the main paper, we accumulate policy gradients along a continuous path from a baseline activation $\mathbf{h}'$ to the original activation $\mathbf{h}$, 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 14 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 improvements on the more challenging AIME2024 benchmark, highlighting its robustness in difficult reasoning scenarios.

Table 14: 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 ↑	AIME2024 ↑
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 influential components and, consequently, more effective intervention. Overall, this ablation confirms that the *integrated* design is a key factor underlying the superior performance of IPG.

Table 15: Accuracy (%) under neuron-space enhancement on Qwen2.5-Math-1.5B-Instruct. The highest accuracy in each column is shown in **bold**.

Method	GSM8K	MATH	AIME	GPQA
Original	82.41	63.00	10.00	24.75
IG	82.94	64.40	13.33	29.29
Neuron Shapley	83.47	64.80	13.33	30.30
AttnLRP	83.62	64.00	16.67	28.28
IPG	**84.61**	**65.80**	**20.00**	**33.33**

D.5 Comparison with Component-Attribution Baselines

We compare IPG with Integrated Gradients (IG), Neuron Shapley, and AttnLRP on Qwen2.5-Math-1.5B-Instruct. For these baselines, we first sample rollouts from the same model and hold the generated tokens fixed during attribution. The scalar attribution objective is the sum of the log probabilities of the answer tokens. The comparison uses the same task-level aggregation, top-neuron selection, and intervention protocol across methods. All interventions are performed in the neuron space, and evaluation uses greedy decoding.

Table 15 shows that each baseline improves accuracy over the original model, indicating that conventional attribution methods also identify components useful for reasoning. IPG achieves the highest accuracy across all four benchmarks. The comparison supports the practical benefit of its outcome-based sequential attribution, although the methods also differ in their attribution objectives.

D.6 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:

$$\text{Dice}(\mathbf{P_A}, \mathbf{P_B}) = \frac{2|\mathbf{P_A} \cap \mathbf{P_B}|}{|\mathbf{P_A}| + |\mathbf{P_B}|}. \tag{28}$$

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.

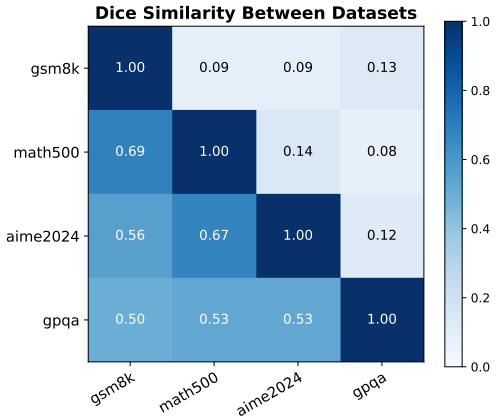


Figure 8: Dice similarity between neuron indices across datasets. We compare neuron indices discovered in 4 different datasets, highlighting shared reasoning structure across tasks, with diagonal showing random baseline.

Top 1 Context (Max Val: 29.6250)

total height of the heels by the total number of heels to get the average height .ٮٮ Let 's start with the first step . We know that there are three women wearing 4 -inch heels and three women wearing 2 -inch heels . So

Top 2 Context (Max Val: 28.8750)

from the amount of money Mike has to find out how much change he will receive .ٮٮ Let 's start with the first step :ٮٮ 1 . Calculate the discount amount :ٮ The original price of the jeans is $ 4 0 , and the discount

Top 3 Context (Max Val: 28.3750)

. Find out how many minutes Peter needs to exercise on Tuesday to reach his goal .ٮٮ Let 's start with the first step :ٮٮ Peter exercised for 1 6 minutes on Monday and 2 3 minutes on Sunday . So , the total time

Feature 486 (layer 20)

Top 1 Context (Max Val: 37.5000)

First , we need to find a common denominator to add these fractions . The common denominator for 4 and 2 is 4 . So , we convert \(\ frac { 1 }{ 2 } \) to a fraction with a denominator

Top 2 Context (Max Val: 36.7500)

6 y \ le q 1 6 \ impl ies y \ le q \ frac { 1 6 }{ 6 } \ approx 2 . 6 7 \ ]ٮ The maximum integer value for \(y \) is 2 .

Top 3 Context (Max Val: 36.0000)

{ d }{ 6 }ٮ \ ]ٮٮ 4 . To add these fractions , find a common denominator . The least common multiple of 1 0 and 6 is 3 0 :ٮ \ [ٮ \ frac { d }{ 1 0

Feature 513 (layer 20)

Figure 9: SAE features 486 and 513 from layer 20 of Qwen2.5-Math-1.5B-Instruct, identified by IPG. Tokens receiving higher activation are shown in darker green.



Feature 12581 (layer 24) Feature 13446 (layer 24)

Figure 10: SAE features 12581 and 13446 from layer 24 of Qwen2.5-Math-1.5B-Instruct, identified by IPG. Tokens receiving higher activation are shown in darker green.



Feature 13499 (layer 26) Feature 24361 (layer 26)

Figure 11: SAE features 13499 and 24361 from layer 26 of Qwen2.5-Math-1.5B-Instruct, identified by IPG. Tokens receiving higher activation are shown in darker green.

Table 16: 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)

D.7 Discussion

D.7.1 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.

Causality-motivation of IPG While standard gradient-based attribution method measures local sensitivity, we argue that our IPG framework is more *causality-motivated* than correlation-based method (e.g. SAE-R) for two reasons. We list the comparison in Table 16.

CMA-style Attribution: IPG is conceptually aligned with 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: We do not rely on attribution alone. The causal link is established in Section 3.2. We empirically validate causal roles of identified components underlying the reasoning mechanism subsequently in Section 4.2.

D.8 Case Studies

D.8.1 Component-Level Disentanglement

In this section, we provide more details and more examples about the semantic contexts that maximally trigger these in reasoning components (Section 4.4.2). We aggregate the textual contexts across the GSM8K dataset that yield the highest activations for our top-ranked, IPG-identified features. We include the results in Figures 9 to 11.

Experiment Setup: We analyze Sparse Autoencoder (SAE) features extracted from the Qwen2.5-Math-1.5B-Instruct model. For each feature index of interest (e.g., those identified by IPG from a given layer), we obtain the total activation counts, peak frequency tokens, and the top-activation values along with their corresponding token contexts (a window of 17 tokens centered on the activation peak). This allows us to qualitatively inspect the behavior of individual SAE features. The experiments are performed on the GSM8K task, focusing on layer {20,24,26}.

D.8.2 Generated Responses before and after Intervention

In this section, we provide more examples with our IPG framework corresponding to Section 3.2. 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.

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.\n\n1. **Initial number of sodas:**\n Ellen starts
with 12 sodas.\n\n2. **Sodas after Ellen shakes 3:**\n Ellen shakes 3 sodas,
so there are \\(12 - 3 = 9\\) sodas left.\n\n3. **Sodas after Ellen takes 1
unshaken soda:**\n Ellen takes 1 unshaken soda, so there are \\(9 - 1 = 8\\)
sodas left, with 3 sodas still shaken.\n\n4. **Sodas after Ellen’s brother
takes 1 shake and 2 unshaken sodas:**\n 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.\n\n5. **Sodas after Ellen’s friend comes
along:**\n Ellen‘s friend now has the remaining 5 sodas, with 2 of them being
shake sodas.\n\nThe 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:\n\\[\n\\textProbability = \\frac\\textNumber of shake
sodas\\textTotal number of sodas = \\frac25\n\\]\n\nTo express this
probability as a percentage, we multiply by 100:\n\\[\n\\frac25 \\times 100 =
40\\%\n\\]\n\n
Thus, the likelihood that Ellen's friend gets sprayed with soda from a shake
can is \\(\\boxed40\\).
```

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.\n\n1. **Initial number of sodas:**\n Ellen starts
with 12 sodas.\n\n2. **Sodas after Ellen shakes 3:**\n Ellen shakes 3 sodas,
so she has 9 sodas left unshaken.\n \\[\n \\textUnshaken sodas = 12 - 3 = 9\n
\\]\n \\[\n \\textShaken sodas = 3\n \\]\n\n3. **Sodas after Ellen takes 1
unshaken soda:**\n Ellen takes 1 unshaken soda for herself, so she has 8
unshaken sodas left.\n \\[\n \\textUnshaken sodas = 9 - 1 = 8\n \\]\n \\[\n
\\textShaken sodas = 3\n \\]\n\n4. **Sodas after Ellen’s brother takes 1
shake and 2 unshake:**\n Ellen‘s brother takes 1 shake soda and 2 unshake
sodas, so the number of sodas changes as follows:\n \\[\n \\textShake sodas =
3 - 1 = 2\n \\]\n \\[\n \\textUnshake sodas = 8 - 2 = 6\n \\]\n\n5. **Sodas
after Ellen’s friend comes along:**\n 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.\n \\[\n \\textTotal
sodas left = 2 + 6 = 8\n \\]\n \\[\n \\textProbability that Ellen’s friend
gets a shake soda = \\frac28 = \\frac14 = 0.25\n \\]\n \\[\n \\textPercentage
= 0.25 \\times 100 = 25 \% \n \\]\n\n Therefore, the likelihood that Ellen's
friend gets sprayed with soda from a shake can is \\(\\boxed25\\).
```

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.\n\n1. **Calculate John‘s regular annual earnings:**\n John makes
$2000 per week. There are 52 weeks in a year.\n \\[\n \\textRegular annual
earnings = 2000 \\times 52 = 104000\n \\]\n\n2. **Calculate the monetary
reward from the award:**\n The award has a 1 time monetary reward of
$5000.\n\n3. **Calculate the total earnings for the year:**\n Add the regular
annual earnings to the monetary reward.\n \\[\n \\textTotal earnings = 104000
+ 5000 = 109000\n \\]\n\n
Therefore, the total amount of money John made that year is
\\(\\boxed109000\\).
```

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.\n\n1. **Calculate John's regular annual earnings:**\n John makes
\$2000 per week. There are 52 weeks in a year.\n \\[\n \\textRegular annual
earnings = 2000 \\times 52 = 104000\n \\]\n\n2. **Calculate the monetary
reward from the award:**\n The award has a 1 time monetary reward of
\$5000.\n\n3. **Calculate the 5\% raise in salary:**\n The 5\% raise in
salary is applied to his regular weekly earnings.\n \\[\n \\textRaise per
week = 2000 \\times 0.05 = 100\n \\]\n Since the raise is applied to each
week, the total raise for the year is:\n \\[\n \\textTotal raise for the year
= 100 \\times 52 = 5200\n \\]\n\n4. **Calculate the total earnings for the
year:**\n Add the regular annual earnings, the monetary reward, and the total
raise.\n \\[\n \\textTotal earnings = 104000 + 5000 + 5200 = 114200\n \\]\n
\GTherefore, the total amount of money John made that year is
\\(\\boxed114200\\)."
```

Case Study on GSM8K for Qwen2.5-Math-1.5B-Instruct. Steering one IPG SAE feature 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.\n\nWe know that 20 cars drove through in the
remaining 15 minutes. \n\nLet‘s denote the number of cars that drove through
in the first 15 minutes as ’x‘. \n\took an exit, we need to subtract these
from the total number of n. Since 5 cars cars to find the number of cars that
actually drove through the traffic jam. \n\nSo, the total number of cars that
drove through the traffic jam is (x + 20) - 5.\n\nWe are given that there
were originally 30 cars on the motorway. \n\nTherefore, (x + 20) - 5 =
30.\n\n Simplifying the equation, we get x + 20 - 5 = 30.\n\nx + 15 = 30.\n\n
Subtracting 15 from both sides, we get x = 15.\n\nSo, the number of cars that
drove through the traffic jam in the first 15 minutes is 15.\n\n\\boxed15
```

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.\n\nWe know that 20 cars drove through in the
remaining 15 minutes. \n\nLet's denote the number of cars that drove through
in the first 15 minutes as 'x'. \n\nSince 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.\n\nWe 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. \n\nSo, we can write the equation:
x + 20 = 25.\n\nTo solve for 'x', we subtract 20 from both sides of the
equation: x = 25 - 20.\n\nx = 5.\n\nHowever, 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. \n\nSince 5 cars took an exit, the number of
cars that were in the first 15 minutes is 30 - 5 = 25. \n\nHowever, we know
that 20 cars drove through in the remaining 15 minutes. \n\nSo, 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.\n\n1. **Total number of people**: There are 11 people in total (5 Democrats, 5 Republicans, and 1 Independent).\n\n2. **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.\n\n3. **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.\n\n4. **Arranging the parties**: There are 3 parties (Democrats, Republicans, and Independent), which can be arranged in 3 = 6 ways around the table.\n\n5. **Arranging the members within each party**: Within each party, the members can be arranged in the following ways:\n - Democrats: 5 ways\n - Republicans: 5 ways\n - Independent: 1 way (since there is only 1 member)\n\n6. **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:\n - Total = 6 * 5 * 5 * 1\n\n7. **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:\n - Total = (6 * 5 * 5 * 1) / 11\n\n8. **Calculating the total**: Now, let's calculate the total number of arrangements:\n - 5 = 5 * 4 * 3 * 2 * 1 = 120\n - Total = (6 * 120 * 120 * 1) / 11\n - Total = (6 * 14400) / 11\n - Total = 86400 / 11\n - Total = 7872\n\nSo, 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:\n\n\\boxed7872".

Enhanced:

To solve this problem, we need to consider the restrictions and the total number of arrangements.\n\n1. 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.\n\n2. 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.\n\n3. 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.\n\n4. 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).\n\n5. 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.\n\n6. Now, we calculate the total number of arrangements: 2 * 5 * 5 = 2 * 120 * 120 = 28800.\n\n7. 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 28800.\n\n\\boxed28800.

Case Study on Math500 for Llama-3.1-8B-Instruct. Steering 5% IPG neurons with scaling factor $\gamma = 1.1$ on layer 22.

Table 17: Performance comparison across models under different methods.

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

D.9 Comparison with Training-based Behavior Control for Long-Horizon Reasoning

Experimental Setup. To demonstrate our IPG is 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 17, 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.10 Behavior-control beyond Reasoning

D.10.1 Reflection Behavior

Experimental Setup. To demonstrate flexibility of 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 18, 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 18: 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.10.2 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 19 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 intervention.

Table 19: 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**

D.11 Scalability of IPG

Beyond our primary evaluations on architectures (e.g., Qwen2.5-Math-1.5B-Instruct and Llama-3-8B-Instruct), we investigate the scalability of IPG by applying it to a larger-scale model, Qwen3-14B. The objective is to verify whether our framework can consistently localize and modulate reasoning-critical components as model capacity and architectural complexity increase.

We apply IPG to identify and intervene on crucial neuron-level components within Qwen3-14B. Using standard greedy decoding, we evaluate our IPG-enhanced model across three challenging reasoning benchmarks: GPQA, AIME, and MATH500.

Table 20: Performance comparison of the baseline Qwen3-14B and the IPG-enhanced model (intervening on neurons) using greedy decoding. Metrics are reported as absolute accuracy (%).

Model Setup	**GPQA**	**AIME**	**MATH500**
Qwen3-14B (Original)	27.27	33.33	73.60
Qwen3-14B (IPG)	**34.34**	**40.00**	**76.00**
Δ *Improvement*	*+7.07*	*+6.67*	*+2.40*

As demonstrated in Table 20, IPG yields consistent performance improvements across all evaluated datasets. These enhancements validate that IPG is not constrained by model size. Regardless of scale, our framework robustly pinpoints the output-oriented mechanisms driving advanced reasoning, enabling reliable, granular capability modulation even in highly parameterized models.

E Use of Large Language Models (LLMs)

We use large language models (LLMs) as auxiliary tools in the writing process, specifically to help identify grammatical errors and polish our writing. Beyond this, LLMs were not involved in core research contributions, including the design of methods, theoretical analysis, and experimental setup. All analyses, and conclusions presented remain the sole responsibility of the authors.

F Broader Impact

We believe there are benefits of advancing interpretable, controllable reasoning, especially given that our method improves existing reasoning models. Our method can have clear societal benefits: more efficient and controllable AI assistants could broaden access to high-quality educational support, scientific reasoning, and problem-solving tools.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes. main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Appendix D.7.1.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: In Appendix A.4.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In Section 4.1 and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We use open-access data set but complete code are currently unavailable.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: In Section 4.1 and Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide confidence interval for greedy decoding which we use in our experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, our research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes. In Appendix F.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not have such risks

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: We use public assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not have new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Yes. In Appendix E.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.