GRAIL: Gradient-Reweighted Advantages for Reinforcement Learning with Verifiable Rewards

Tej Deep Pala [1] **Vernon Toh** [1] **Soujanya Poria** [1]

Abstract

Reinforcement learning with verifiable rewards (e.g., GRPO) is now a common way to improve mathematical reasoning in large language models (LLMs). However, current methods usually broadcast one sequence-level advantage to all tokens, or use costly process reward models (PRMs) for step-level supervision. Uniform advantage distribution assumes that all tokens contribute equally to the final reward. This dilutes the gradient signal, since flawed reasoning steps and filler words are updated as strongly as valid logical inferences. To address this, we introduce Gradient-Reweighted Advantage (GRAIL), an intrinsic token-wise advantage reweighting method. GRAIL uses gradient-activation saliency to place more weight on tokens that are more locally sensitive to the final answer. Evaluations across five models from the Qwen3, R1-distilled and OctoThinker families show that GRAIL consistently outperforms GRPO. GRAIL achieved an average improvement of 3.60% in accuracy and 3.05% in Pass@3, demonstrating that fine-grained reasoning alignment can be achieved without process-level supervision.

1. Introduction

Enhancing mathematical reasoning capabilities in large language models (LLMs) has become a critical research frontier. For domains where correctness can be programmatically verified, reinforcement learning with verifiable rewards has emerged as a primary paradigm. Traditionally, alignment has relied on algorithms like PPO (Schulman et al., 2017) to optimise reward models based on human feedback; however, the focus has recently shifted toward more memory-efficient alternatives that utilise outcome-based reward signals to cultivate complex reasoning behaviours without necessitating a separate critic model.

Recent literature highlights the efficacy of this outcome-driven approach. Group Relative Policy Optimization (GRPO) introduced a framework by normalising rewards within a group of sampled completions (Shao et al., 2024). At scale, GRPO-style training has produced strong chain-of-thought reasoning models, such as DeepSeek-R1 (Guo et al., 2025) and Qwen-Math (Yang et al., 2024). Subsequent studies have sought to refine this approach; for instance, DAPO addresses instabilities from entropy collapse (Yu et al., 2026), while BNPO (Xiao et al., 2025) and Dr. GRPO (Liu et al., 2025) propose alternative normalisation strategies to mitigate biases from variable completion lengths.

Despite these advances, credit assignment across the reasoning trace remains coarse. Current GRPO-style methods usually broadcast the same sequence-level advantage to all tokens in a completion. This assumes that all tokens contribute equally to the final outcome. As a result, the gradient signal is diluted. Flawed reasoning steps and filler words can receive updates as strong as critical logical steps. We hypothesise that reweighting the token-level advantage with an intrinsic saliency signal can focus the update and improve reasoning alignment.

To address this limitation, this paper introduces Gradient-Reweighted Advantage (GRAIL), an intrinsic token-wise advantage reweighting method. GRAIL computes a gradient-activation saliency score for each token. It then uses this score to reweight the sequence-level advantage in the policy-gradient loss. Extensive evaluations across five models from the Qwen3, R1-distilled and OctoThinker families show that GRAIL consistently outperforms the GRPO baseline, with an improvement of 3.60% in average accuracy and 3.05% in Pass@3.

2. Related Work

2.1. Process Supervision and Reward Models

Outcome reward models (ORMs) provide a single binary signal at the end of a completion, offering no guidance on

[1]Nanyang Technological University. Correspondence to: Soujanya Poria <soujanya.poria@ntu.edu.sg>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

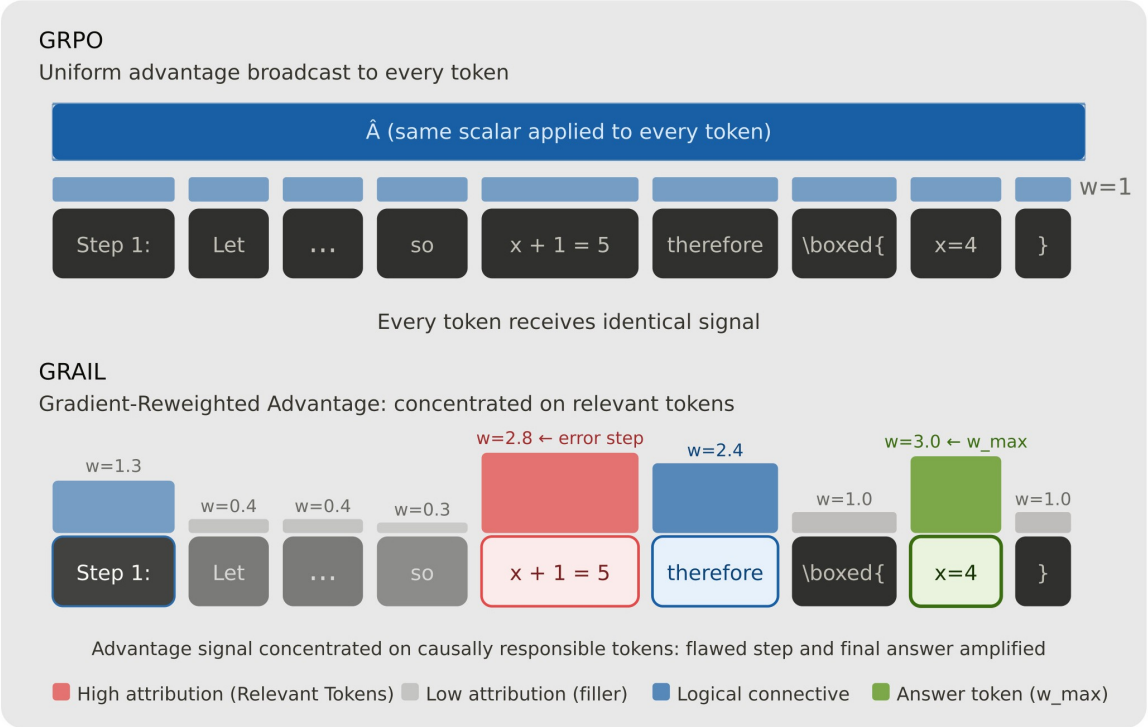

Figure 1. Comparison of credit assignment in GRPO versus GRAIL. **Top:** Standard GRPO broadcasts a uniform scalar advantage across all tokens in a rollout. **Bottom:** GRAIL uses gradient-activation saliency to reweight token-level advantages. High-saliency tokens receive stronger updates, while incidental filler words receive weaker updates.

which reasoning steps were correct or incorrect. Process reward models (PRMs) address this by assigning a reward to each intermediate reasoning step, providing a denser supervision signal that can identify where in a chain-of-thought an error was introduced. While PRMs trained on human labels (Lightman et al., 2024) or via automated verification like Math-Shepherd (Wang et al., 2024) outperform ORMs on complex benchmarks, they remain difficult to scale. A key limitation of PRM-based approaches is that they require extensive human annotation or rely on trained verifiers, and automatically synthesised data often yields inferior performance (Zhang et al., 2025). In contrast, GRAIL derives token-level weights directly from the model's intrinsic gradients during training, providing fine-grained credit assignment without the overhead of external supervision.

2.2. Gradient-Based Saliency

Saliency methods estimate which input features are most important for a model output. Gradient-based methods use the gradient of the output with respect to the input as a local sensitivity signal (Baehrens et al., 2010; Simonyan et al., 2013; Selvaraju et al., 2020). The element-wise gradient $\times$ activation estimator used by GRAIL is a first-order Taylor-style approximation (Baehrens et al., 2010). It is cheaper than path-integral methods such as Integrated Gradients (Sundararajan et al., 2017).

In language models, saliency methods are often used after training to identify influential tokens (Bastings & Filippova, 2020; Ferrando et al., 2022) or to trace factual knowledge to model components (Meng et al., 2022). Attention weights are also used as an importance proxy, but they are not a reliable indicator of causal effect (Clark et al., 2019; Jain & Wallace, 2019; Wiegreffe & Pinter, 2019). Using saliency during training is less explored. Some token-MDP methods assign per-step rewards by marginalising over future trajectories (Arjona-Medina et al., 2019; Pignatelli et al., 2023; Chan et al., 2024). GRAIL is different. It uses gradient-based saliency to reweight the existing GRPO objective, without changing the reward function or rollout procedure.

2.3. Fine-Grained Credit Assignment and Advantage Reweighting

To address the coarse nature of sequence-level advantage in algorithms like GRPO, recent works have explored mechanisms to redistribute credit at the token level. Several approaches use intrinsic uncertainty metrics, such as token entropy, to scale advantages. For instance, Cheng et al. (2026) amplify advantages on high-entropy tokens to encourage exploration, while Chen et al. (2025) segment generations by entropy and reweight advantages based on the overlap of low-entropy segments between correct and incorrect rollouts. Other heuristics include upweighting critical

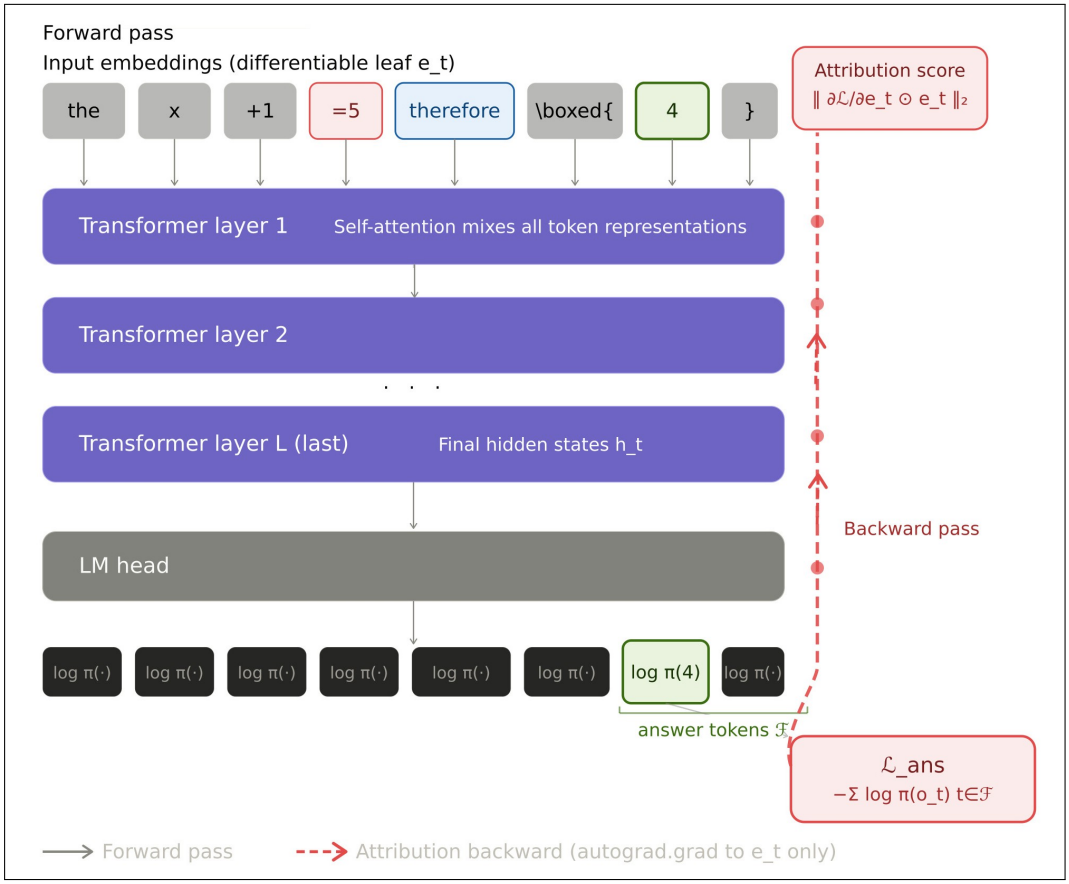

Figure 2. The gradient-activation saliency mechanism in GRAIL. To compute the token-specific saliency score, the differentiable leaf is placed at the input embeddings ($\mathbf{e}_t$). The gradient of the answer loss ($\mathcal{L}_{\text{ans}}$) flows back through the sequence-wide self-attention layers. This gives a local sensitivity signal for earlier reasoning tokens with respect to the final answer tokens.

tokens based on attention dynamics (Li et al., 2025) or reshaping advantages using a combination of response-level confidence and token-level logit certainty (Xie et al., 2025).

Moving beyond heuristic proxies, PRIME (Cui et al., 2025) attempts to tackle the credit assignment problem by training an implicit PRM online using only outcome labels, generating dense token-level rewards that are aggregated into the Monte Carlo advantage estimate. Closely related to our work is Outcome-grounded Advantage Reshaping (OAR) (Li et al., 2026), which redistributes advantages using a bi-level gating mechanism based on token influence. To approximate this influence, OAR's gradient-based variant (OAR-G) calculates the KL divergence between a clean outcome distribution and a noisy outcome distribution derived by injecting Gaussian noise into the embeddings. Consequently, OAR requires two forward passes (one clean, one noisy) and one backward pass per sequence to compute the self-distillation gradient. In contrast, GRAIL directly leverages the analytic gradient of the final answer loss ($\mathcal{L}_{\text{ans}}$) with respect to the input embeddings. This fundamental formulation difference allows GRAIL to extract a precise attribution signal using only a single forward and a single backward pass, eliminating the computational overhead of noise injection and auxiliary self-distillation objectives. Furthermore, GRAIL introduces targeted structural interventions to ensure valid weighting at delimiter and EOS tokens, ensuring a stable and mathematically direct attribution link between intermediate reasoning and the final outcome.

3. Methodology

Reinforcement learning from verifiable rewards usually spreads one advantage over the whole generated sequence. This ignores that some reasoning tokens matter more than others. Our goal is to make this credit assignment less noisy. In this section, we introduce Gradient-Reweighted Advantage (GRAIL), an intrinsic token-wise advantage reweighting method. GRAIL uses gradient-activation saliency to reweight the sequence-level advantage before applying the policy-gradient update.

3.1. Token-Wise Advantage Reweighting

Consider a language model policy parameterised by θ, denoted as π_θ. Given a query q, the model generates an output

consisting of a sequence of tokens $o = \{o_1, o_2, \ldots, o_T\}$. In the context of Group Relative Policy Optimization (GRPO), a group of rollouts G is generated for the same query, and an outcome-based reward r_i is computed to derive a relative advantage estimate $\hat{A}_i$ for each rollout o_i. In our experiments, we use the Dr. GRPO style of advantage computation:

$$A_i = r_i - \frac{1}{|G|} \sum_{i=1}^{|G|} r_i$$

To simplify notation, we define the importance sampling ratio of the active policy to the reference policy at timestep t as $\rho_{i,t}(\theta)$:

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(o_{i,t} \mid q, o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t} \mid q, o_{i,<t})} \tag{1}$$

The standard policy gradient objective for GRPO (excluding the KL divergence penalty) is formulated as:

$$\mathcal{L}_{\text{GRPO}} = -\frac{1}{|G|} \sum_{i=1}^{|G|} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min \Big(\rho_{i,t}(\theta) \hat{A}_i, \\ \text{clip}\big(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon\big) \hat{A}_i \Big)$$

A limitation of this formulation is the identical use of the sequence-level advantage $\hat{A}_i$ across all tokens at timestep t. Under this paradigm, a filler word or a flawed intermediary derivation that coincidentally precedes a correct answer receives the same gradient update as the correct logical inferences that actively drove the positive outcome. The model is effectively unable to distinguish which specific parts of its reasoning chain to correct or reinforce.

To resolve this, we introduce a token-specific saliency weight, $w_{i,t}$. This weight is a local proxy for how sensitive the final answer is to token $o_{i,t}$. We integrate it directly into the policy-gradient objective:

$$\mathcal{L}_{\text{GRAIL}} = -\frac{1}{|G|} \sum_{i=1}^{|G|} \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \min \Big(\rho_{i,t}(\theta) \text{sg}(w_{i,t}) \hat{A}_i, \\ \text{clip}\big(\rho_{i,t}(\theta), 1-\epsilon, 1+\epsilon\big) \text{sg}(w_{i,t}) \hat{A}_i \Big)$$

3.2. Gradient-Activation Saliency

We need a token-specific saliency score to compute $w_{i,t}$. We use a gradient-activation saliency method. To get a non-zero signal for intermediate reasoning tokens, we compute the gradient with respect to a representation before the transformer body. This lets the backward pass propagate through all causal attention layers. We therefore use the input embeddings, $\mathbf{e}_t$, as the differentiable leaf.

Let $\mathcal{F}$ denote the subset of positional indices corresponding to the final answer tokens (e.g., the tokens constrained within a `\boxed...` final answer marker). The outcome-based loss, $\mathcal{L}_{\text{ans}}$, is defined as the negative log-likelihood of these answer tokens:

$$\mathcal{L}_{\text{ans}} = -\sum_{t \in \mathcal{F}} \log \pi_\theta(o_t \mid o_{<t})$$

The gradient flow from the answer loss backward to the input embedding of an arbitrary token t is continuously mediated by the self-attention mechanism across all L layers of the network. For any preceding reasoning token $r \notin \mathcal{F}$, the gradient $\frac{\partial \mathcal{L}_{\text{ans}}}{\partial \mathbf{e}_r}$ evaluates to a non-zero vector exclusively when the attention mechanism has routed information from position r forward into the final hidden states of the answer tokens, $\mathbf{h}_{\mathcal{F}}^{(L)}$.

We define the raw saliency score, s_t, for each token as the L_2-norm of the element-wise product between the gradient of the answer loss with respect to its input embedding and the embedding itself:

$$s_t = \left\| \frac{\partial \mathcal{L}_{\text{ans}}}{\partial \mathbf{e}_t} \odot \mathbf{e}_t \right\|_2$$

Empirically, the raw saliency scores s_t are non-negative and exhibit a heavy-tailed distribution. Consequently, a small subset of tokens yields scores several orders of magnitude larger than the bulk of the sequence. Applying standard z-score normalisation directly to these raw scores is suboptimal, as the resulting weights would be disproportionately dominated by extreme outliers, leading to erratic gradient updates.

To mitigate this and ensure a stable learning signal, we project the saliency scores into log-space prior to standardisation. We define the log-score $\tilde{s}_t$ as:

$$\tilde{s}_t = \log(s_t + \epsilon)$$

where ϵ is a small constant added to ensure numerical stability. The final saliency weight w_t is then computed by standardising these log-scores and scaling them to a controlled distribution:

$$w_t = w_{\text{mean}} + \frac{\tilde{s}_t - \mu_{\log}}{\sigma_{\log}} \cdot \sigma_w$$

$\mu_{\log}$ and $\sigma_{\log}$ denote the mean and standard deviation of the log-scores across all completion tokens in the rollout. The

hyperparameter w_{mean} establishes the baseline weight for a neutral token, while σ_w controls the spread and intensity of the saliency reweighting.

Finally, to guarantee training stability and prevent any single high-saliency token from overwhelmingly dominating the policy gradient, the derived weights are strictly clipped to a bounded range $[w_{\min}, w_{\max}]$.

This gives a first-order local sensitivity signal for each token with respect to the final answer. The score is computed from the current model during training. It does not require external heuristics or an independently trained process reward model.

Furthermore, the saliency weights are strictly wrapped in a stop-gradient operator, $\text{sg}(\cdot)$. This is essential because the weights themselves depend on θ through the saliency computation. Without the stop-gradient, the optimiser would encounter an unaccounted second-order term, $\frac{\partial w_t}{\partial \theta}$, rendering the objective ill-defined as a standard policy gradient estimator. By treating the weights as fixed scalars from the optimiser's perspective, $\mathcal{L}_{\text{GRAIL}}$ remains a valid, importance-weighted policy gradient.

3.3. Simple Weight Corrections

A few token types need simple corrections. First, the final answer tokens often receive low saliency scores, because their gradient signal is mostly self-referential. To keep final correctness important, we set the weights at these inner answer positions to the maximum bound, $w_{\max}$. In contrast, final answer delimiters, such as the opening `\boxed{`, can receive high scores because they are close to the answer. We do not want to reinforce formatting over reasoning, so we assign delimiter positions a neutral weight of w_{mean}.

Finally, tokens generated after the final answer, including the EOS token, have near-zero saliency. They do not have a differentiable path to the answer loss under the causal attention mask. If we suppress these updates, the model may become less certain about when to stop. We therefore reset all post-answer positions to w_{mean}.

4. Experimental Setup

To rigorously evaluate the efficacy of Gradient-Reweighted Advantage (GRAIL) against Group Relative Policy Optimization (GRPO) baseline, we design a comprehensive experimental framework targeting mathematical reasoning capabilities.

4.1. Training

Our experiments are conducted across multiple architectures to ensure the generalisability of the proposed saliency mechanism. Specifically, we utilise five models drawn from three distinct families: Qwen3 (4B and 8B) (Team, 2025), Deepseek-R1-Distill-Llama-8B (Guo et al., 2025) and OctoThinker (OctoThinker-3B-Short and OctoThinker-8B-Short) (Wang et al., 2025). All models are then trained on the DeepMath-103K dataset (He et al., 2025), which provides a robust and diverse corpus of verifiable mathematical problems suitable for outcome-based reinforcement learning.

To test the impact of our loss reweighting formulation, the base models, datasets, and core training infrastructure remain identical across both the GRAIL and GRPO runs. A comprehensive breakdown of all training hyperparameters, including learning rates, global batch sizes, and GRAIL hyperparameters, is detailed in Appendix A.

4.2. Evaluation

We evaluate the generalisation of mathematical problem-solving capabilities across six established benchmarks encompassing varying degrees of difficulty: Math500 (Lightman et al., 2024), AIME 2024 (Zhang & Math-AI, 2024), AMC 2023 (math ai, 2025), MinervaMath (Lewkowycz et al., 2022), CollegeMath (Tang et al., 2024), and OlympiadBench (He et al., 2024).

During inference, we adopt a consistent sampling strategy across all evaluation datasets. For each problem, we sample three distinct generations using a sampling temperature of 0.6. A list of evaluation hyperparameters is provided in Appendix A. Final model performance is reported as the pass rate and average accuracy across the three generations. This ensures that the reported metrics reflect the model's reliable reasoning capacity.

5. Results

In this section, we evaluate the efficacy of Gradient-Reweighted Advantage (GRAIL) across five model architectures and six mathematical reasoning benchmarks. The empirical results demonstrate that dynamically redistributing the gradient signal to influential tokens consistently elevates problem-solving capabilities, yielding an average improvement of 3.60% in accuracy and 3.05% in Pass@3 over the GRPO baseline.

5.1. Main Results

Table 1 details the performance of the Qwen3, R1-Distill and OctoThinker model families across the evaluation suite. Overall, GRAIL yields an impressive average absolute improvement of approximately 3.60% in Average Accuracy and 3.05% in Pass@3 across all evaluated models.

Most notably, GRAIL drives substantial gains on the most challenging benchmarks that require extended, complex reasoning chains. On the AIME 2024 benchmark, Qwen3-

Model	MATH500	AMC23	AIME24	College MATH	Olympiad Bench	Minerva MATH	Average
				Average Accuracy			
Qwen3-4B							
GRPO	88.07	77.50	27.78	52.17	45.83	39.83	55.20
GRAIL	91.47	80.00	36.67	53.17	47.50	44.36	**58.86** (+3.66)
Qwen3-8B							
GRPO	90.07	77.50	36.67	54.00	46.67	41.42	57.72
GRAIL	92.07	82.50	47.78	56.00	49.17	44.73	**62.04** (+4.32)
OctoThinker-3B							
GRPO	43.87	12.50	1.11	28.17	15.00	14.95	19.27
GRAIL	46.40	26.67	3.33	28.83	16.00	16.05	**22.88**(+3.61)
OctoThinker-8B							
GRPO	48.53	22.50	2.22	29.00	14.67	21.08	23.00
GRAIL	54.53	23.33	4.44	33.00	20.50	25.12	**26.82** (+3.82)
R1-Distill-Llama-8B							
GRPO	81.00	71.67	21.11	42.50	42.17	30.64	48.18
GRAIL	82.60	81.67	25.56	41.67	42.00	30.88	**50.73** (+2.55)
				Pass@3			
Qwen3-4B							
GRPO	93.60	92.50	43.33	55.50	53.50	47.06	64.25
GRAIL	93.80	97.50	53.33	55.00	54.00	47.79	**66.90** (+2.65)
Qwen3-8B							
GRPO	95.00	90.00	50.00	57.00	52.50	45.96	65.08
GRAIL	95.40	87.50	60.00	59.00	53.00	48.89	**67.29** (+2.21)
OctoThinker-3B							
GRPO	59.60	22.50	3.33	38.50	26.00	24.63	29.09
GRAIL	61.40	40.00	6.67	39.00	27.00	24.63	**33.12** (+4.03)
OctoThinker-8B							
GRPO	65.20	32.50	3.33	38.50	25.50	31.99	32.84
GRAIL	68.40	40.00	10.00	43.50	29.50	36.40	**37.97** (+5.13)
R1-Distill-Llama-8B							
GRPO	89.00	85.00	36.67	50.00	48.50	42.28	58.57
GRAIL	91.00	90.00	36.67	49.50	50.00	41.54	**59.79** (+1.22)

Table 1. Average accuracy and Pass@3 (%) across six mathematical reasoning benchmarks. Results are evaluated at a sampling temperature of 0.6 and averaged over three generations per problem. Across all evaluated models, GRAIL consistently achieves higher overall reasoning performance than the GRPO baseline.

4B improves its accuracy from 27.78% to 36.67%, while Qwen3-8B leaps from 36.67% to 47.78%. Similarly, OctoThinker-8B doubles its accuracy (2.22% to 4.44%) and triples its Pass@3 rate (3.33% to 10.00%) on the same benchmark. The Qwen3-8B architecture exhibits the largest overall benefit in average accuracy, recording a 4.32% absolute increase, while OctoThinker-8B sees the highest Pass@3 increase at 5.13%. This suggests that the gradient-activation saliency mechanism is highly effective at unlocking reasoning bottlenecks regardless of the model's base reasoning capacity.

Compared to GRPO, the targeted reinforcement in GRAIL successfully amplifies critical logical steps across the entirety of the evaluation suite without any task-specific degradation. Whether evaluated on complex, multi-step competition mathematics (e.g., AIME and AMC) or advanced curriculum-level benchmarks, intrinsic loss reweighting consistently elevates performance. This confirms that the method generalises well across different mathematical problem-solving distributions.

To verify the statistical significance of the observed performance gains, we conducted a Wilcoxon signed-rank test on the paired average accuracy scores between the GRPO baseline and GRAIL across all evaluated models and benchmarks. The test shows a statistically significant improvement when using GRAIL ($W = 459.5$, $p = 1.3 \times 10^{-8}$). This suggests that the gains are unlikely to come from random variation alone.

These findings show that token-wise advantage reweighting

Model	MATH500	AMC23	AIME24	College MATH	Olympiad Bench	Minerva MATH	Average
Qwen3-4B							
OAR-G	89.73	81.67	30.00	53.50	46.67	42.77	57.39
GRAIL	91.47	80.00	36.67	53.17	47.50	44.36	**58.86** (+1.47)
Qwen3-8B							
OAR-G	89.27	74.17	32.22	53.33	46.67	38.85	55.75
GRAIL	92.07	82.50	47.78	56.00	49.17	44.73	**62.04** (+6.29)

Table 2. Performance comparison between GRAIL and OAR-G. GRAIL achieves higher average accuracy across both architectures, with a +6.29% absolute improvement on the Qwen3-8B model. The substantial gains on long-horizon tasks like AIME24 demonstrate that GRAIL's direct analytic gradient approach isolates critical reasoning steps more effectively than OAR's noise-injected sensitivity proxy.

can improve reasoning alignment without step-level supervision.

5.2. Comparison with OAR-G

Table 2 details the performance of GRAIL and OAR-G when applied to the Qwen3-4B and Qwen3-8B architectures. GRAIL consistently outperforms OAR-G across the evaluation suite. On the 4B model, GRAIL yields a 1.47% absolute improvement in average accuracy. Crucially, as the base model scales to 8B parameters, the performance delta widens dramatically. GRAIL achieves a 62.04% average accuracy on Qwen3-8B compared to OAR-G's 55.75%, representing a 6.29% absolute gain.

The superiority of GRAIL is most pronounced on highly complex, multi-step derivation tasks. On AIME 2024, GRAIL increases accuracy by 6.67% on the 4B model and 15.56% on the 8B model relative to OAR-G. While OAR-G relies on injecting Gaussian noise into embeddings to approximate token influence via a self-distillation objective, this proxy signal can become misaligned with strict verifier acceptance on long reasoning traces. The empirical gap between the two methods suggests that deriving the attribution signal directly from the analytic gradient of the answer loss ($\mathcal{L}_{\text{ans}}$) provides a sharper and more reliable attribution between intermediate logic and final mathematical correctness.

5.3. Ablations

5.3.1. Differentiable Leaf Placement

To validate the architectural design of the gradient-activation saliency, we conduct an ablation on the placement of the differentiable leaf. Table 3 details the performance of Qwen3-4B when the saliency score is derived from various depths within the model: the input embeddings, and the first, middle and penultimate hidden layers.

Performance drops as the differentiable leaf is moved deeper into the network. Deriving saliency scores at the input embeddings yields the highest average accuracy (58.15%). In contrast, truncating the backward pass by placing the leaf at the penultimate layer results in the lowest performance (53.41%), alongside substantial drops on complex benchmarks like AIME24 (declining from 36.67% to 24.44%).

This steady degradation confirms our theoretical hypothesis regarding credit assignment. To compute useful weights for earlier reasoning tokens, the gradient must fully propagate backwards through the entire sequence-wide attention mechanism. Bypassing these layers strips the gradient of its structural context, rendering the resulting saliency weights less effective for policy optimisation.

5.3.2. Symmetric vs. Asymmetric Weighting

To evaluate the impact of saliency weighting on different reward polarities, we ablate the application of GRAIL across correct and incorrect completions. Table 4 compares the performance of Qwen3-4B and Qwen3-8B when the saliency signal is applied symmetrically (to all rollouts) versus asymmetrically (exclusively to correct or wrong rollouts).

Applying the weights only to wrong rollouts gives the highest peak accuracy: 58.86% on the 4B model and 62.04% on the 8B model. In particular, this asymmetric penalty drives gains on complex tasks like AIME24 (reaching 47.78% for Qwen3-8B). Conversely, applying the weights only to correct rollouts consistently underperforms, indicating that positive reinforcement alone is insufficient for effective credit assignment.

The symmetric version is the cleanest formulation. However, the strong result of the "wrong rollouts only" setting is informative. This empirical result suggests that GRAIL is more effective at correcting logical errors than it is at reinforcing correct steps. As corroborated by our positional analysis (Section 5.4), correct rollouts produce a diffuse saliency signal across intermediate steps, whereas flawed derivations in wrong rollouts generate sharp, highly concentrated saliency weights. This suggests that negative reinforcement is especially useful for correcting flawed reasoning paths.

Differentiable Leaf Layer	MATH500	AMC23	AIME24	College MATH	Olympiad Bench	Minerva MATH	Average
Input Embeddings	88.67	81.67	36.67	51.83	48.17	41.91	58.15
First Hidden Layer	88.47	79.17	28.89	51.83	47.17	40.56	56.01
Middle Hidden Layer	89.13	76.67	26.67	52.33	45.33	40.20	55.05
Penultimate Layer	86.47	72.50	24.44	51.83	44.50	40.69	53.41

Table 3. Ablation on the differentiable leaf placement for GRAIL using Qwen3-4B. Deriving saliency scores from the input embeddings yields the highest average accuracy. Placing the leaf at progressively deeper intermediate layers steadily degrades performance, confirming that gradient propagation must capture all layers to assign credit to earlier reasoning tokens accurately.

GRAIL Application Strategy	MATH500	AMC23	AIME24	College MATH	Olympiad Bench	Minerva MATH	Average
Qwen3-4B							
Correct Rollouts Only	85.47	71.67	25.56	54.67	44.00	38.60	53.33
Wrong Rollouts Only	91.47	80.00	36.67	53.17	47.50	44.36	**58.86**
All Rollouts	88.67	81.67	36.67	51.83	48.17	41.91	58.15
Qwen3-8B							
Correct Rollouts Only	88.13	83.33	35.56	52.50	47.17	42.52	58.20
Wrong Rollouts Only	92.07	82.50	47.78	56.00	49.17	44.73	**62.04**
All Rollouts	90.67	77.50	38.89	53.00	50.17	42.52	58.79

Table 4. Ablation on symmetric versus asymmetric application of GRAIL weights. Applying the saliency signal exclusively to wrong rollouts yields the highest peak accuracy across both the 4B and 8B architectures. Applying GRAIL symmetrically across all rollouts provides a competitive and more stable learning signal.

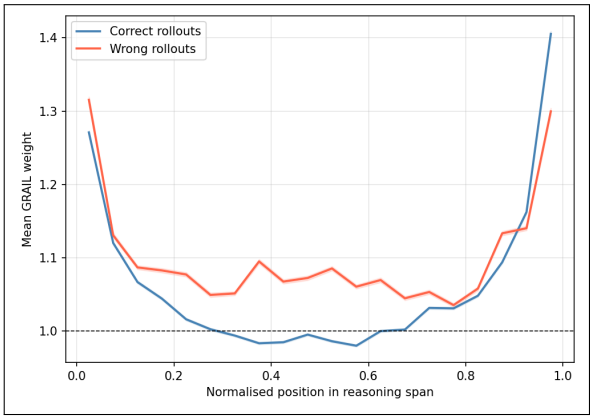

Figure 3. Mean GRAIL weights across normalised positions within the reasoning span.

5.4. Positional Analysis of Saliency Weights

To understand how GRAIL distributes credit across the reasoning chain, we analyse the spatial distribution of saliency weights. For each completion, we normalise the positional indices of the reasoning span to a continuous range of $[0, 1]$. We then aggregate the mean GRAIL weight at each normalised position across the dataset, explicitly segregating the trajectories into correct and wrong rollouts.

The resulting distribution in Figure 3 has a clear "U-shape" across completions. Saliency weights peak significantly at the beginning (positions < 0.1) and the end (positions > 0.9) of the reasoning span. This boundary concentration is consistent with the primacy-recency bias documented in transformer attention: Liu et al. (2024) show that language models disproportionately attend to tokens at the boundaries of long contexts, with middle tokens receiving systematically weaker attention density. Under the causal attention mask, initial premise tokens are attended to by every subsequent token in the sequence, while final concluding tokens directly precede the answer markers that define $\mathcal{L}_{\text{ans}}$. Both positions thus accumulate high gradient signal regardless of their semantic content, producing the observed boundary peaks.

The more diagnostically significant difference lies in the intermediate reasoning span (normalised positions 0.2 to 0.8). For correct rollouts, the mean GRAIL weight drops to, or slightly below, the neutral baseline ($w \approx 1.0$), producing a diffuse and approximately uniform advantage signal. This is consistent with the intuition that a correct derivation distributes its logical contribution broadly, as no single intermediate step is markedly more responsible for the final answer than others. In contrast, wrong rollouts consistently maintain a higher mean weight across this entire intermediate span. We hypothesise that this concentration reflects locations of logical failure: when an error is introduced mid-derivation, the gradient of $\mathcal{L}_{\text{ans}}$ with respect to the input embeddings becomes concentrated on the tokens surrounding that failure, since the attention mechanism routes information from that erroneous step into all subsequent hidden states and ultimately into the answer tokens. This asymmetry could explain the empirical finding from Section 5.3.2, where penalising wrong rollouts with the GRAIL signal is substantially more effective than reinforcing correct ones,

since the saliency weights are most discriminative precisely where they are most needed.

6. Conclusion

In this work, we addressed the limitation of uniform credit assignment in Group Relative Policy Optimization (GRPO) for mathematical reasoning. We hypothesised that broadcasting a sequence-level advantage equally to all tokens dilutes the policy gradient, failing to distinguish between critical logical inferences and flawed intermediate derivations. To resolve this, we introduced Gradient-Reweighted Advantage (GRAIL), a loss reweighting mechanism that derives token-specific importance weights. Extensive evaluations across the five models demonstrate that GRAIL consistently elevates reasoning performance, yielding substantial gains on complex, long-horizon tasks. Crucially, as revealed by our positional analysis, GRAIL achieves this by assigning higher weights to intermediate tokens linked to logical errors. This focuses the learning signal where the policy needs correction.

GRAIL provides a self-contained way to improve fine-grained reasoning alignment in large language models. While currently evaluated on mathematical problem-solving, the nature of this intrinsic gradient-weighting mechanism carries broad implications for the verifiable reinforcement learning paradigm. Future work may explore the application of this saliency framework beyond mathematics, extending its utility into other domains that require verifiable trajectories. GRAIL grounds the weighting signal in the model's own gradients, which makes it a simple path for improving verifiable RL.

Limitations

While Gradient-Reweighted Advantage (GRAIL) provides an effective intrinsic learning signal, the gradient-activation product relies on a first-order approximation of influence. This approach successfully captures the linear sensitivity of the final answer to each token's input representation, but it does not account for higher-order interactions, such as tokens that only become critical in combination with others or counterfactual contributions. Although advanced saliency methods like Integrated Gradients could provide a more theoretically rigorous estimator, implementing them during continuous reinforcement learning would require multiple forward and backward passes per rollout, making them computationally prohibitive.

References

Arjona-Medina, J. A., Gillhofer, M., Widrich, M., Unterthiner, T., Brandstetter, J., and Hochreiter, S. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., and Müller, K.-R. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(61):1803–1831, 2010. URL http://jmlr.org/papers/v11/baehrens10a.html.

Bastings, J. and Filippova, K. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 149–155, 2020.

Chan, A. J., Sun, H., Holt, S., and Van Der Schaar, M. Dense reward for free in reinforcement learning from human feedback. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Chen, X., Li, X., Sun, Z., and Yu, W. Beyond high-entropy exploration: Correctness-aware low-entropy segment-based advantage shaping for reasoning llms. *arXiv preprint arXiv:2512.00908*, 2025.

Cheng, D., Huang, S., Zhu, X., Dai, B., Zhao, X., Zhang, Z., and Wei, F. Reasoning with exploration: An entropy perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30377–30385, 2026.

Clark, K., Khandelwal, U., Levy, O., and Manning, C. D. What does BERT look at? an analysis of BERT's attention. In Linzen, T., Chrupała, G., Belinkov, Y., and Hupkes, D. (eds.), *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828/.

Cui, G., Yuan, L., Wang, Z., Wang, H., Zhang, Y., Chen, J., Li, W., He, B., Fan, Y., Yu, T., et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

Ferrando, J., Gállego, G. I., and Costa-jussà, M. R. Measuring the mixing of contextual information in the transformer. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 8698–8714, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.595. URL https://aclanthology.org/2022.emnlp-main.595/.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

He, C., Luo, R., Bai, Y., Hu, S., Thai, Z., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., et al. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, 2024.

He, Z., Liang, T., Xu, J., Liu, Q., Chen, X., Wang, Y., Song, L., Yu, D., Liang, Z., Wang, W., et al. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning. *arXiv preprint arXiv:2504.11456*, 2025.

Hugging Face. Math-verify. https://github.com/huggingface/Math-Verify, 2024. Accessed: 2025-11-25.

Jain, S. and Wallace, B. C. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3543–3556, 2019.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Li, Y., Dong, Z., Sun, Y., Wang, W., Xiong, S., Luo, Y., Liu, J., Lu, H., Wang, J., Su, W., et al. Attention illuminates llm reasoning: The preplan-and-anchor rhythm enables fine-grained policy optimization. *arXiv preprint arXiv:2510.13554*, 2025.

Li, Z., Kang, L., Xiao, F., Xing, L., Si, Q., Li, Z., Gong, W., Yang, D., Xiao, Y., and Guo, H. Outcome-grounded advantage reshaping for fine-grained credit assignment in mathematical reasoning. *arXiv preprint arXiv:2601.07408*, 2026.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.

Liu, Z., Chen, C., Li, W., Qi, P., Pang, T., Du, C., Lee, W. S., and Lin, M. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

math ai. amc23: A hugging face dataset. https://huggingface.co/datasets/math-ai/amc23, 2025. Accessed: 2026-05-26.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt. *Advances in neural information processing systems*, 35:17359–17372, 2022.

Pignatelli, E., Ferret, J., Geist, M., Mesnard, T., van Hasselt, H., Pietquin, O., and Toni, L. A survey of temporal credit assignment in deep reinforcement learning. *arXiv preprint arXiv:2312.01072*, 2023.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: visual explanations from deep networks via gradient-based localization. *International journal of computer vision*, 128(2):336–359, 2020.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International conference on machine learning*, pp. 3319–3328. PMLR, 2017.

Tang, Z., Zhang, X., Wang, B., and Wei, F. Mathscale: Scaling instruction tuning for mathematical reasoning. *arXiv preprint arXiv:2403.02884*, 2024.

Team, Q. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., and Gallouédec, Q. TRL: Transformers Reinforcement Learning, 2020. URL https://github.com/huggingface/trl.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Wang, Z., Zhou, F., Li, X., and Liu, P. Octothinker: Mid-training incentivizes reinforcement learning scaling. *arXiv preprint arXiv:2506.20512*, 2025. Preprint.

Wiegreffe, S. and Pinter, Y. Attention is not not explanation. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 11–20, 2019.

Xiao, C., Zhang, M., and Cao, Y. Bnpo: Beta normalization policy optimization. *arXiv preprint arXiv:2506.02864*, 2025.

Xie, C., Pan, R., Wu, X., Zhang, Y., Fu, J., Gao, T., and Zhou, G. Unlocking exploration in rlvr: Uncertainty-aware advantage shaping for deeper reasoning. *arXiv preprint arXiv:2510.10649*, 2025.

Yang, A., Zhang, B., Hui, B., Gao, B., Yu, B., Li, C., Liu, D., Tu, J., Zhou, J., Lin, J., et al. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Zhang, Y. and Math-AI, T. American invitational mathematics examination (aime) 2024, 2024.

Zhang, Z., Zheng, C., Wu, Y., Zhang, B., Lin, R., Yu, B., Liu, D., Zhou, J., and Lin, J. The lessons of developing process reward models in mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10495–10516, 2025.

A. Hyperparameters

A.1. Training Hyperparameters

Hyperparameter	Value
Optimization & Training	
Precision	bfloat16
Attention Implementation	Flash Attention 2
Optimizer	Fused AdamW
Learning Rate	1.0×10^{-6}
Learning Rate Scheduler	Constant
Warmup Steps	20
Total Train Batch Size	256
Max Steps	200
Max Gradient Norm	1.0
Random Seed	42
GRPO & Rollout Generation	
Loss Type	dr_grpo
Group Size ($\|G\|$ / num_generations)	8
GRPO Iterations (num_iterations)	4
KL Penalty (β)	0.0
Clip Range (ϵ)	0.2
Upper Clip Range (ϵ_{high})	0.28
Max Completion Length	4096
Sampling Temperature	1.0
Top-p	1.0
Top-k	50
Repetition Penalty	1.0
Mask Truncated Completions	True
Gradient-Reweighted Advantage (GRAIL)	
Neutral Baseline Weight (w_{mean})	1.0
Standard Deviation (σ_w)	0.5
Minimum Weight Bound ($w_{\min}$)	0.5
Maximum Weight Bound ($w_{\max}$)	5.0

Table 5. Hyperparameters used for training Qwen3 and OctoThinker models via the Hugging Face trl library.

All models are fine-tuned using the open-source trl library from Hugging Face (von Werra et al., 2020). The GRPO objective is configured to utilize the Dr. GRPO loss formulation (dr_grpo), with the KL divergence penalty (β) set to 0. A comprehensive list of the hyperparameters used across our training runs is provided in Table 5.

A.2. Evaluation Hyperparameters

Hyperparameter	Value
Number of Generations	3
Sampling Temperature	0.6
Top-p	0.95
Top-k	20

Table 6. Sampling hyperparameters utilized during the benchmark evaluation phase.

During the evaluation phase across all six mathematical reasoning benchmarks, we maintain a consistent sampling and verification protocol to ensure robust and fair comparisons. For each problem, the model generates multiple candidate solutions. The final mathematical answer is extracted from the reasoning trace by parsing the contents of the \boxed{}. We then utilize the math-verify library to evaluate the mathematical equivalence between the extracted prediction and the dataset's ground truth, which accounts for varied algebraic and symbolic representations (Hugging Face, 2024).

The specific sampling parameters used to generate the evaluation rollouts are detailed in Table 6.

A.3. Prompt

To elicit structured, step-by-step reasoning and ensure the final answers are reliably formatted for automated extraction, we wrap all mathematical queries using a standardized zero-shot template. During both the training rollout generation and the final evaluation phases, the input query q is formatted as follows:

```
{Question}. Please reason step by step, and put your final answer within
\boxed{}.
```

This explicit structural constraint guarantees that the generated reasoning traces are compatible with the math-verify extraction pipeline, while also providing a consistent delimiter for the GRAIL mechanism to anchor its positional weight corrections.

B. Computational Efficiency and Training Time

Calculating the intrinsic token-level weights in GRAIL introduces a computational overhead. Specifically, deriving the saliency scores requires an additional backward pass through the attention layers to the input embeddings prior to the policy update step.

To quantify this overhead, we report the total wall-clock training time for both the standard GRPO baseline and GRAIL across our evaluated models. All training runs were conducted on a single compute node equipped with 4 $\times$ NVIDIA H200 GPUs. As detailed in Table 7, GRAIL introduces an average computational overhead of approximately 50% to 60% over GRPO.

Model	GRPO	GRAIL	Overhead
OctoThinker-3B	1h 05m	1h 40m	+54%
Qwen3-4B	1h 41m	2h 30m	+49%
OctoThinker-8B	2h 05m	3h 20m	+60%
Qwen3-8B	2h 30m	4h 00m	+60%

Table 7. Comparison of total wall-clock training time between standard GRPO and GRAIL. All models were trained for 200 steps on 4 $\times$ NVIDIA H200 GPUs.