COUNTERFACTUAL CREDIT ASSIGNMENT FOR POLICY OPTIMIZATION

Mykola Khandoga, Rui Yuan, Vinay Kumar Sankarapu
Lexsi Labs
mykola.khandoga@lexsi.ai

ABSTRACT

Policy gradient methods for language model reasoning, such as GRPO and DAPO, assign uniform credit to all generated tokens - the filler phrase "Let me think" receives the same gradient update as the critical calculation "$23 + 45 = 68$." We propose counterfactual importance weighting: mask reasoning spans, measure the drop in answer probability, and upweight tokens accordingly during policy gradient updates. Our method requires no auxiliary models or external annotation, instead importance is estimated directly from the policy model's own probability shifts. Experiments on GSM8K across three models spanning the Qwen and Llama families demonstrate consistent improvements over uniform baselines and faster convergence to equivalent accuracy. Inverting the importance signal hurts performance, confirming we capture genuine causal structure rather than noise. Analysis shows the method correctly prioritizes calculation steps over scaffolding text. We view these findings as establishing counterfactual importance weighting as a foundation for further research rather than a complete solution.

1 INTRODUCTION

The advancements in reinforcement learning with verifiable rewards (RLVR) and the GRPO algorithm (Shao et al., 2024) along with its successors - DAPO (Yu et al., 2025), Dr. GRPO (Liu et al., 2025), and others - have enabled rapid improvements in model capabilities on tasks like mathematics and coding. When GRPO proposed eliminating the critic model, it led to substantial savings in computation and VRAM. But this simplification came at a cost: the granularity of credit assignment was reduced. Now all tokens in a generated solution receive exactly the same reward signal, regardless of their actual contribution to the final answer.

There have been attempts to address this problem through various means. Process Reward Models (PRMs) (Lightman et al., 2023; Wang et al., 2024) provide step-level supervision by training a separate model to score intermediate reasoning steps - but this requires either expensive human annotation or synthetic labels from a stronger model. Token-level reward methods like TLCR (Yoon et al., 2024) train discriminators to assign per-token rewards, adding architectural complexity. Segment-based approaches like SPO (Guo et al., 2025) partition sequences at fixed intervals, but this does not adapt to the actual structure of reasoning.

We propose to use the model's intrinsic signals to determine which parts of the reasoning trace matter most for achieving the correct answer. This is achieved through counterfactual importance: we mask candidate reasoning spans and measure the resulting drop in answer probability. Spans whose removal damages the answer are deemed important and receive higher weight during training. The method requires no auxiliary models and no external annotation. We describe the approach in detail in Section 3.

Research Question: Can we improve GRPO training by assigning higher credit to tokens that causally contribute to correct answers?

We address this question by introducing **counterfactual importance weighting** for GRPO-like training algorithms. Our method identifies reasoning spans (arithmetic expressions, intermediate calculations) and measures their causal importance by masking each span and observing the drop in

answer probability. Spans whose removal significantly damages the answer receive higher importance scores, which then weight the corresponding tokens during gradient computation.

We focus on mathematical reasoning, where task success is determined by a single verifiable answer. This creates a natural 'answer sink' - a concentrated verification signal that makes counterfactual importance estimation well-defined: masking a critical calculation step substantially damages the probability of reaching the correct answer. We hypothesize that domains with more distributed correctness criteria (e.g., code generation, where multiple structural elements must simultaneously be correct) may benefit less from span-level interventions. We test this hypothesis in Appendix B.

To validate that our importance scores capture meaningful signal rather than noise, we compare four weighting strategies:

- **Counterfactual:** Weight tokens by their measured causal importance
- **Inverted:** Weight tokens inversely to importance (control)
- **Random:** Assign random weights (control)
- **Uniform:** Standard DAPO with equal weights (baseline)

If counterfactual importance captures genuine signal, we expect: Counterfactual $>$ Uniform $\geq$ Random $>$ Inverted. Our experiments generally confirm this ordering across multiple models.

Contributions.

1. We propose a counterfactual method for measuring token-level importance in reasoning chains, based on answer probability drop under span masking.
2. We integrate this importance signal into DAPO training through multiplicative token weighting.
3. We provide systematic experiments across model scales (1.7B–3B parameters) showing consistent improvements of **0.8–1.1** percentage points over baseline DAPO on GSM8K.
4. We validate the importance signal through ablations: inverted weighting consistently hurts performance, confirming the method captures genuine causal structure.
5. We provide a detailed empirical analysis of what makes reasoning spans important: calculation chains are $11\times$ enriched in critical spans, 3.5% of spans are distractors that hurt performance, and counterfactual weighting concentrates $1.6\times$ more gradient mass on high-importance tokens. These findings characterize the structure of reasoning traces and may inform future credit assignment methods.

We view this work as a proof of concept: demonstrating that counterfactual importance provides consistent, directional signal for credit assignment. The modest but reliable gains (0.8–1.1 pp) and strong ablation results suggest this is a promising direction, with room for future work on stronger interventions, better span detection, and combination with complementary methods.

2 Related Work

Counterfactual methods in deep learning. Our approach extends a rich lineage of "mask and measure" methods: from occlusion sensitivity in vision (Zeiler & Fergus, 2014), through representation erasure in NLP (Li et al., 2016), to causal tracing in transformers (Meng et al., 2022). In RL, counterfactual credit assignment (Foerster et al., 2018; Mesnard et al., 2021) isolates each action's causal contribution. VinePPO (Kazemnejad et al., 2024) showed that most reasoning tokens do not affect problem-solving probability—directly motivating our work. A detailed historical survey is provided in Appendix C.

Token-level methods for LLM training. Recent work has pursued finer-grained credit assignment through various proxies: Process Reward Models (Lightman et al., 2023) provide step-level supervision but evaluate correctness rather than causal contribution; TLCR trains discriminators for token-level rewards; RTO (Zhong et al., 2024) extracts implicit token rewards from DPO probability ratios; SPO (Guo et al., 2025) partitions sequences into segments with Monte Carlo advantage

estimation. SCAR (Cao et al., 2025) applies Shapley values to distribute rewards among tokens - the closest existing work to explicit counterfactual analysis.

The gap we address. Despite this convergent trajectory, no existing method directly asks: *"What would the outcome be if this token were different?"* Current approaches estimate importance through learned discrimination, probability ratios, or game-theoretic values - but not through intervention. Our counterfactual importance method completes this arc: we adapt the mask-and-measure methodology from interpretability to provide **training signals** for policy optimization, using the same interventional logic that revealed which pixels matter for image classification to identify which tokens matter for reasoning.

Concurrent work. Concurrent work by Ruan et al. (2025) proposes Critical Token Fine-Tuning (CFT), which identifies critical tokens for supervised fine-tuning by checking whether alternative token choices preserve answer correctness. Our work addresses a complementary setting: credit assignment in reinforcement learning with policy gradients. Where CFT produces a binary mask (train or skip), we compute continuous importance weights based on answer probability drop, enabling finer-grained gradient modulation. Where CFT requires full regeneration to verify correctness, our span-level masking requires only forward passes. The convergent finding - that counterfactual reasoning identifies tokens that matter - suggests this is a general principle applicable across training paradigms.

3 METHOD

3.1 BACKGROUND: DAPO

We build on DAPO (Yu et al., 2025), which extends GRPO with decoupled clipping and token-level policy gradient loss. For a prompt x with G sampled completions $\{y_1, \ldots, y_G\}$ and binary rewards $\{r_1, \ldots, r_G\}$, the advantage for completion i is:

$$A_i = \frac{r_i - \mu_r}{\sigma_r + \epsilon} \tag{1}$$

where μ_r and σ_r are the group mean and standard deviation. The policy gradient loss is:

$$\mathcal{L}_{\text{DAPO}} = -\frac{1}{\sum_i T_i} \sum_{i=1}^{G} \sum_{t=1}^{T_i} A_i \cdot \log \pi_\theta(y_t^{(i)} | x, y_{<t}^{(i)}) \tag{2}$$

Critically, the advantage A_i is uniform across all tokens in completion i.

We implement the DAPO algorithm using HuggingFace TRL's `loss_type="dapo"` option, which includes token-level loss normalization and decoupled clipping (Clip-Higher) as described by Yu et al. (2025). Dynamic sampling is not used. For clarity, we present the unclipped token-sum form above. Our contribution is the addition of per-token importance weights w_t.

Intuition: Addressing gradient dilution. In standard GRPO-like algorithms, the policy gradient averages over all tokens equally, effectively "diluting" the learning signal with updates from filler phrases, boilerplate setup ("Let me solve this step by step..."), and redundant restatements. Our weighting scheme concentrates gradient mass on tokens with demonstrated causal influence, namely those whose removal damages answer probability. This can be viewed as a non-parametric alternative to Process Reward Models: where PRMs require training an auxiliary value function $V(s)$ to estimate step-wise contributions (Lightman et al., 2023), counterfactual masking provides direct estimates of causal importance using only the policy model itself, with no additional parameters or training data. The connection to Mesnard et al. (2021)'s "skill vs. luck" decomposition is direct: by isolating tokens that *caused* success rather than merely co-occurred with it, we bias gradient updates toward the reasoning steps that actually determined the outcome. Our analysis confirms this intuition: calculation chains receive $11\times$ higher importance than scaffolding tokens (Table 5), and 3.5% of spans are outright distractors whose removal *improves* answer probability—content that uniform weighting would incorrectly reinforce.

3.2 COUNTERFACTUAL IMPORTANCE ESTIMATION

We measure token importance through causal intervention. Let a completion y be decomposed as $y = (r, a)$, where r is the reasoning prefix and a is the final answer span.

Answer-probability drop. For each detected reasoning span $s_k \subset r$, we define the counterfactual *drop* as:

$$D(s_k) = \log P_\theta(a \mid x, r_{-s_k}) - \log P_\theta(a \mid x, r) \tag{3}$$

where r_{-s_k} denotes the reasoning prefix with span s_k replaced by a fixed placeholder string. More negative $D(s_k)$ indicates higher causal importance: masking the span damages answer probability. Positive $D(s_k)$ indicates a *distractor* span whose removal actually *improves* answer probability.

We convert this into a non-negative *importance score* for weighting:

$$I(s_k) = -D(s_k) \tag{4}$$

High $I(s_k)$ indicates the span is causally necessary for the model to produce the correct answer.

Computing $P_\theta(a \mid x, r)$. We compute $\log P_\theta(a \mid x, r)$ by teacher forcing over the answer tokens:

$$\log P_\theta(a \mid x, r) = \sum_{t=1}^{|a|} \log \pi_\theta(a_t \mid x, r, a_{<t}) \tag{5}$$

All counterfactual forward passes are run under `torch.no_grad()` to avoid memory overhead.

Placeholder design. We use replacement rather than deletion to preserve positional structure and avoid length-change confounds. The placeholder token is the model's pad token (or end-of-text token if pad is unavailable), repeated to match the span length.

Why spans, not tokens? We operate at span granularity rather than per-token for two reasons. First, computational cost: per-token masking would require 200–500 forward passes per completion (one per token), whereas span-level masking requires only 5–10, reducing overhead by 20–50×. Second, semantic coherence: individual tokens lack standalone meaning - "23", "+", "45" separately do not constitute a reasoning step, whereas the span "23 + 45 = 68" represents a complete arithmetic operation.

Span detection. We identify reasoning spans via pattern matching: arithmetic expressions (e.g., `23 + 45 = 68`), intermediate calculations, and sentence boundaries. We process up to $K_{\max} = 10$ spans per completion.

Weight assignment. For each token t within span s_k, we assign importance weight proportional to $I(s_k)$. To ensure weights remain bounded, we first normalize importance scores within each completion to $[0, 1]$:

$$\hat{I}(s_k) = \frac{I(s_k) - \min_j I(s_j)}{\max_j I(s_j) - \min_j I(s_j) + \epsilon} \tag{6}$$

then map to the weight range $[w_{\min}, w_{\max}]$:

$$w_t = w_{\min} + \hat{I}(s_k) \cdot (w_{\max} - w_{\min}) \tag{7}$$

Tokens outside detected spans receive baseline weight $w_t = 1$. Final answer tokens receive a fixed boost ($w_{\text{ans}} = 1.5$).

3.3 IMPORTANCE-WEIGHTED POLICY GRADIENT

We modify the DAPO loss by introducing per-token importance weights:

$$\mathcal{L}_{\text{CF-DAPO}} = -\frac{1}{\sum_i T_i} \sum_{i=1}^{G} \sum_{t=1}^{T_i} w_t^{(i)} \cdot A_i \cdot \log \pi_\theta(y_t^{(i)} | x, y_{<t}^{(i)}) \tag{8}$$

This formulation upweights gradient contributions from high-importance tokens (critical calculations) while downweighting filler tokens.

3.4 ABLATION CONDITIONS

To validate that our importance scores capture genuine signal, we compare four weighting modes:

Mode	Weight Assignment
Counterfactual	$w_t \propto \hat{I}(s_k)$ for tokens in span s_k (normalized importance)
Inverted	$w_t \propto (1 - \hat{I}(s_k))$ - high-importance spans receive *low* weight
Random	$w_t \sim \text{Uniform}(w_{\min}, w_{\max})$, independently per token
Vanilla	$w_t = 1$ for all t (standard DAPO, uniform credit)

If counterfactual importance captures meaningful signal, we expect: Counterfactual $>$ DAPO $\geq$ Random $>$ Inverted.

4 EXPERIMENTS

4.1 SETUP

Models. We evaluate on three base models spanning different architectures and scales: Qwen3-1.7B, Qwen2.5-3B, and Llama3.2-3B. All models are trained with LoRA adapters (Hu et al., 2022) ($r = 32$, $\alpha = 32$) for parameter efficiency.

Dataset. We train and evaluate on GSM8K (Cobbe et al., 2021), a dataset of 7,473 grade-school math problems requiring multi-step arithmetic reasoning. We evaluate on the standard 1,319-problem test set using exact-match accuracy with numeric parsing.

Training. We use DAPO with $G = 8$ completions per prompt, learning rate 2.5×10^{-5}, batch size 16 with 4 gradient accumulation steps, for 500 gradient steps. We use temperature 0.6 and top-p 0.95 for generation. Each configuration is run with 3 random seeds.

Reward. We use binary outcome reward: $r = 1$ if the extracted numeric answer matches the gold answer, $r = 0$ otherwise.

Weighting Hyperparameters. For counterfactual and ablation conditions, we use weight range $[w_{\min}, w_{\max}] = [0.5, 4.0]$, answer boost $w_{\text{ans}} = 1.5$, and maximum $K_{\max} = 10$ spans per completion.

4.2 MAIN RESULTS

Results are summarized in Tables 1, 2, 3 and Figure 1. We observe a consistent pattern across all three model families:

(1) Counterfactual weighting improves over baseline. Counterfactual importance weighting outperforms vanilla DAPO on all models: $+0.9$ pp on Qwen3-1.7B, $+1.1$ pp on Qwen2.5-3B, and $+0.8$ pp on Llama3.2-3B. The improvement is statistically significant for Qwen2.5-3B ($p = 0.002$) at step 500; AUC improvements are significant for both Qwen models ($p < 0.05$). The effect is consistent throughout training (Figure 1), not just at convergence.

(2) Inverted weighting underperforms. Inverting the importance signal - upweighting tokens that don't matter and downweighting those that do - consistently yields worse performance. On Qwen3-1.7B, inverted underperforms DAPO by 1.5 pp; on Llama3.2-3B the gap widens to 1.8 pp. This directional validation confirms the importance signal captures genuine causal structure rather than acting as arbitrary regularization.

Table 1: GSM8K test accuracy (%) at step 500. Mean ± std over seeds. Best in **bold**.

Model	CF	DAPO	Random	Inverted
Qwen3-1.7B	**84.3±0.5**	83.4±0.6	82.5±0.5	81.9±0.9
Qwen2.5-3B	**86.7±0.2**	85.6±0.2	85.8±0.8	85.3±0.3
Llama3.2-3B	**78.9±0.3**	78.2±0.6	78.1±0.6	76.4±1.5

Table 2: GSM8K AUC (%), measuring cumulative accuracy across training. Mean ± std over seeds. Best in **bold**.

Model	CF	DAPO	Random	Inverted
Qwen3-1.7B	**83.2±0.4**	82.5±0.5	82.1±0.5	81.0±0.8
Qwen2.5-3B	**85.8±0.4**	85.3±0.4	85.0±0.8	84.8±0.3
Llama3.2-3B	**77.3±0.3**	76.6±0.6	75.9±0.5	75.6±1.2

(3) Random weighting is broadly neutral. Random weights perform comparably to or slightly below vanilla baseline, confirming that non-uniform weighting alone is insufficient - the *direction* of importance matters. Counterfactual weighting, which assigns high weights to causally important spans, yields consistent gains across all seeds and models.

(4) Results hold across architectures. Counterfactual weighting consistently outperforms all baselines across both Qwen and Llama model families, suggesting the method generalizes beyond a single architecture.

(5) Improved sample efficiency. Beyond final accuracy, counterfactual weighting accelerates learning throughout training. On average, CF reaches DAPO's final accuracy (at step 500) substantially earlier: at step 100 for Qwen3-1.7B, step 125 for Qwen2.5-3B, and step 225 for Llama3.2-3B. This improved sample efficiency can offset the 32–74% per-step overhead: to reach a given target accuracy, CF often requires comparable or less total wall-clock time than vanilla DAPO.

4.3 Analysis

We analyze counterfactual importance scores across 13,737 spans from 1,467 completions. Beyond validating our method, this analysis reveals structural properties of reasoning traces—what content is critical, what is scaffolding, and what is actively harmful—that may inform credit assignment methods beyond our specific approach.

Importance distribution. The distribution of importance drops is heavily left-skewed (skewness $= -2.19$, kurtosis $= 7.73$), indicating that most spans have moderate importance while a minority are critical. Table 4 shows the breakdown: only 10.9% of spans are "critical" (drop < -500), while 45.4% fall in the "moderate" range. Notably, 3.5% of spans (481 total) are distractors: masking them actually improves answer probability, suggesting they add noise rather than signal.

What content is critical? We analyze textual patterns that distinguish critical spans (drop < -500, N=1,500) from low-importance spans ($-50 <$ drop < 0, N=1,349). Table 5 reveals striking differences: **calculation chains** (spans containing sequential equations like "$x = y = z$") are **11.2**× more prevalent in critical spans than low-importance ones. Multiplication and division operations show **6.5**× enrichment, while proportion and rate reasoning shows **3.9**× enrichment. Conversely, equation *setup* ("let x denote...") is **depleted** in critical spans (0.37×) - such spans are necessary scaffolding but not causally important for reaching the answer.

Token-level weight distribution. Across 858,798 non-padding tokens, the weight range $[0.5, 4.0]$ is fully utilized: 1.2% of tokens receive minimum weight (0.5×) while 21.8% receive maximum weight (4.0×, including answer tokens). This confirms the method produces substantial differentiation rather than near-uniform weights.

Table 3: Statistical significance of CF vs. DAPO (paired t-test across seeds). $^*p < 0.05$.

Model	**Δ@500**	p**@500**	**ΔAUC**	p**AUC**
Qwen3-1.7B	+0.90%	0.112	+0.74%	0.035*
Qwen2.5-3B	+1.09%	0.002*	+0.46%	0.049*
Llama3.2-3B	+0.76%	0.125	+0.46%	0.613

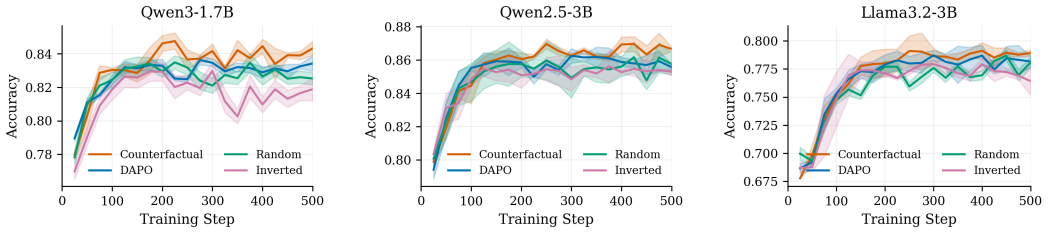

Figure 1: Training curves on GSM8K for Qwen3-1.7B (left), Qwen2.5-3B (middle), and Llama3.2-3B (right). Counterfactual weighting (red) consistently outperforms vanilla DAPO (blue) throughout training. Inverted weighting (purple) underperforms, validating that importance direction matters. Shaded regions show ± 1 std across seeds.

Gradient concentration. We quantify how counterfactual weighting redistributes gradient mass compared to uniform weighting. Across 858,798 tokens, we categorize by normalized importance: high ($\hat{I} > 0.8$), medium ($0.5 < \hat{I} \leq 0.8$), and low ($\hat{I} \leq 0.5$). Under uniform weighting, gradient contribution is proportional to token count. Under CF weighting, high-importance tokens (26.3% of tokens) receive 42.5% of gradient mass - a **1.6× concentration**. Conversely, low-importance tokens (53.9% of tokens) receive only 32.2% of gradient mass - a **0.6× dilution**. This represents a 2.7× shift in relative gradient allocation from filler tokens to causally important reasoning steps, empirically validating the "gradient dilution" intuition: CF weighting focuses parameter updates on the tokens that actually determine task success.

Additional analysis of span position, length correlations, distractor spans, and a qualitative example are provided in Appendix D.

Summary. Our analysis reveals that counterfactual importance successfully identifies reasoning structure: calculation chains and arithmetic operations are critical (11× and 6.5× enriched); setup and headers are scaffolding (0.4–0.6× depleted); and 3.5% of spans are actively harmful distractors. The method captures causal importance rather than surface features, though length and position correlate as expected. These findings support using counterfactual importance to focus gradient updates on the spans that actually determine task success.

5 Limitations

Computational overhead. The method requires additional forward passes to estimate span importance, adding 32–74% overhead depending on the number of spans per completion. While the optimizations discussed below (batched masking, caching, early termination) mitigate this cost, the overhead remains substantial for large-scale training. The improved sample efficiency can offset this cost, but practitioners must weigh per-step overhead against convergence benefits.

Domain specificity. Our span detection relies on regex patterns tuned for arithmetic reasoning: equations, calculations, and sentence boundaries. This works well for GSM8K but fails to capture domain-relevant structure in other settings. The null results on MBPP+ (Appendix B) demonstrate this limitation—conditionals, loops, and function definitions are not detected as spans. Extending the method to new domains requires designing appropriate span detectors, which may require domain expertise.

Table 4: Distribution of importance drops across 13,737 spans. More negative indicates higher importance.

Category	Drop Range	Count	%
Critical	< -500	1,500	10.9%
Important	$[-500, -200)$	4,176	30.4%
Moderate	$[-200, -50)$	6,231	45.4%
Low	$[-50, 0)$	1,349	9.8%
Distractor	≥ 0	481	3.5%

Table 5: Content pattern prevalence in critical vs. low-importance spans. Enrichment > 1 indicates the pattern is more common in critical spans; < 1 indicates depletion.

Pattern	Critical	Low	Enrich.
Calculation chain ($= \cdots =$)	28.3%	2.5%	**11.2**×
Multiply/divide operations	41.9%	6.4%	**6.5**×
Proportion/rate reasoning	22.1%	5.6%	**3.9**×
Total/sum operations	48.5%	17.0%	2.9×
Conclusion ("therefore...")	16.8%	7.1%	2.4×
Equation setup ("let...")	3.7%	10.1%	0.37×
Step headers ("Step 1:")	1.7%	2.8%	0.62×

Imperfect causal intervention. Span masking is an approximation to true counterfactual reasoning. Replacing tokens with placeholders may introduce distributional shift: the model has never seen padding tokens mid-sequence during pretraining, so its behavior under masking may not reflect "what would happen if this information were absent." Alternative interventions (e.g., resampling from the model's own distribution, or activation patching) could provide cleaner causal estimates but at higher computational cost.

6 Conclusion

We introduced counterfactual importance weighting for GRPO-like family of algorithms. The method identifies causally important reasoning spans by measuring answer probability drop under masking, then uses this signal to reweight token-level policy gradients. Experiments across three model families demonstrate consistent improvements of 0.8–1.1 percentage points over baseline DAPO, with faster convergence to equivalent accuracy. While these gains are modest, they are consistent across architectures, validated by ablations, and achieved with no auxiliary models—suggesting that counterfactual signal is both real and usable. Analysis of 13,737 spans reveals the method correctly identifies calculation chains as critical (11× enriched) while deprioritizing scaffolding tokens, concentrating 1.6× more gradient mass on high-importance reasoning steps.

Beyond the training improvements, our analysis reveals structural properties of reasoning traces that may be independently useful. The finding that 3.5% of spans are outright distractors—content whose removal improves answer probability—suggests that standard uniform-credit training actively reinforces harmful content. The 11× enrichment of calculation chains in critical spans, versus 0.37× depletion of equation setup, quantifies the intuition that "not all tokens matter equally" and provides a basis for future work on adaptive curricula or data filtering.

The approach requires no auxiliary models, no external annotation, and adds only forward-pass overhead - providing a non-parametric alternative to Process Reward Models that extracts credit assignment signal directly from the policy model's own probability estimates. The key validation is directional: inverting the importance signal hurts performance, confirming that the method captures genuine causal structure rather than acting as arbitrary regularization.

However, the method does not transfer to code generation (Appendix B), where correctness is distributed across multiple structural elements rather than concentrated in a single answer. Future work should explore domain-specific span detection for code (e.g., control flow, function boundaries) and computational optimizations to reduce per-step overhead.

REFERENCES

Meng Cao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. SCAR: Shapley credit assignment for more efficient RLHF. *arXiv preprint arXiv:2505.20417*, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, 2018.

Yuejiang Guo et al. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. *arXiv preprint arXiv:2505.23564*, 2025. NeurIPS 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Sarthak Jain and Byron C Wallace. Attention is not explanation. In *Proceedings of NAACL-HLT*, pp. 3543–3556, 2019.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Porber, Alessandro Sordoni, Nicolas Le Roux, Mohammad Ghodsi, and Sarath Chandar. VinePPO: Unlocking rl potential for llm reasoning through refined credit assignment. *arXiv preprint arXiv:2410.01679*, 2024.

Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Zichen Liu, Changyu Liu, Wenda Zheng, Chao Zhang, Junhui Lin, Shelby Heinecke, Caiming Xiong, Silvio Savarese, and Steven Hoi. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 17359–17372, 2022.

Thomas Mesnard, Theophane Weber, Fabio Viola, Shantanu Thakoor, Alaa Saade, Anna Harutyunyan, Will Dabney, Tom Stepleton, Nicolas Heess, Arthur Guez, et al. Counterfactual credit assignment in model-free reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 7654–7664, 2021.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2009.

Zhiwen Ruan, Yixia Li, He Zhu, Yun Chen, Peng Li, Yang Liu, and Guanhua Chen. Enhancing large language model reasoning via selective critical token fine-tuning. *arXiv preprint arXiv:2510.10974*, 2025.

Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701, 1974.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International Conference on Machine Learning (ICML)*, pp. 3319–3328, 2017.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 12388–12401, 2020.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Eunseop Yoon, Hee Suk Yoon, SooHwan Eom, Gunsoo Han, Daniel Wontae Nam, Daejin Jo, Kyoung-Woon On, Mark A Hasegawa-Johnson, Sungwoong Kim, and Chang D Yoo. Tlcr: Token-level continuous reward for fine-grained reinforcement learning from human feedback. *arXiv preprint arXiv:2407.16574*, 2024.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision (ECCV)*, pp. 818–833. Springer, 2014.

Han Zhong, Gao Feng, Wei Xiong, Qingfeng Zhao, et al. DPO meets PPO: Reinforced token optimization for RLHF. *arXiv preprint arXiv:2404.18922*, 2024. ICML 2025 Spotlight.

A Computational Overhead Analysis

Our counterfactual importance method introduces computational overhead from the additional forward passes required to estimate token importance. We provide a detailed analysis of this overhead and discuss both current optimizations and potential improvements.

Sources of overhead. For each completion in a training batch, we must compute the answer probability drop under masking for every reasoning span. This requires one additional forward pass per span. Table 6 summarizes the overhead across our experimental configurations.

Table 6: Computational overhead of counterfactual importance estimation. Runtime measured on a single L40S GPU. Spans/completion reflects spans passing importance thresholds during training.

Model	DAPO (min)	CF (min)	OH	Spans/ Comp.	CF/ Step
Qwen3-1.7B	405	535	+32%	5.2	21.5s
Llama3.2-3B	403	578	+43%	7.2	21.0s
Qwen2.5-3B	412	715	+74%	5.8	36.4s

The overhead varies significantly across models (32–74%), primarily driven by the cost of forward passes on each architecture rather than the number of spans detected. Counterfactual estimation consumes 30–42% of total training time, comparable to the generation phase itself. For Qwen2.5-3B, time per step breaks down as: generation (42.4s, 49%), log-probs (1.2s, 1%), CF estimation (36.4s, 42%), and other operations (5.9s, 7%).

Memory overhead. Interestingly, counterfactual estimation does not increase GPU memory usage. Since CF forward passes only compute log-probabilities without gradient tracking, peak memory remains dominated by the backward pass of the main training loop. We observe slightly *lower* memory usage in CF runs (11.3–12.5 GB) compared to vanilla (12.6–13.3 GB), likely due to more aggressive memory clearing between CF passes.

Current optimizations. Our implementation includes several optimizations:

- **Gradient-free forward passes**: CF estimation uses `torch.no_grad()`, reducing per-pass cost to approximately 0.4–0.7× a standard forward pass (0.37s vs 0.75s for Llama3.2-3B).
- **Hash-based caching**: We cache importance weights keyed by (prompt, completion) hash to avoid recomputation for identical samples across epochs. However, due to on-policy sampling generating new completions each step, cache hit rates remain below 1% in practice.
- **Batched span detection**: Reasoning spans are detected via vectorized regex matching over the full batch before any forward passes, minimizing Python overhead.

Potential improvements. Several optimizations could substantially reduce overhead in future work:

1. **Batched CF forward passes**: Currently, masked sequences are processed with variable-length inputs. Padding and batching CF passes could improve GPU utilization by 2–3×.
2. **Proxy model estimation**: Using a smaller model (e.g., Qwen-0.5B) to estimate importance weights for a larger model being trained. Initial experiments suggest importance rankings transfer reasonably across model scales.
3. **Sparse span selection**: Rather than evaluating all spans, select only the top-k longest or most syntactically complex spans. With $k = 3$, this could reduce CF passes by 40–60% with minimal signal loss.
4. **Importance weight caching across epochs**: For offline RL settings where the dataset is fixed, importance weights could be precomputed once and reused, amortizing the overhead to near-zero for subsequent epochs.
5. **Asynchronous computation**: CF estimation for batch t could overlap with gradient computation for batch $t-1$ using separate CUDA streams, hiding latency on multi-GPU setups.
6. **Early termination**: Skip CF computation for completions where all candidate answers receive zero reward (uniformly incorrect), as their loss contribution is already zero in DAPO.

Cost-benefit analysis. Despite the 32–74% overhead, counterfactual weighting improves learning efficiency as measured by accuracy gain per gradient step. The practical question is whether CF's per-step gains justify its slower iterations:

- **Fixed step budget**: For 500 training steps, CF takes 32–74% longer but achieves 0.8–1.1pp higher final accuracy - a favorable tradeoff when final performance matters more than iteration speed.
- **Fixed compute budget**: Given the same wall-clock time, vanilla DAPO completes more steps but CF still matches or exceeds its accuracy due to more efficient credit assignment per step.

We recommend vanilla DAPO for rapid hyperparameter search, with CF weighting applied for final training runs where accuracy improvements justify additional compute.

B Code Generation: A Negative Result

We evaluated counterfactual weighting on MBPP+ code generation to test whether the method generalizes beyond mathematical reasoning. Table 7 shows results across training checkpoints.

We find no significant improvement over baseline DAPO - at the best checkpoint, vanilla slightly outperforms CF (54.2% vs 53.7%). Early training shows a small CF advantage, but vanilla takes over from step 150 onward. Results are largely within noise (±1–2%).

We hypothesize this null result stems from the *distributed* nature of code correctness. In math, a single numeric answer serves as the verification target: masking a calculation chain destroys the

Table 7: MBPP+ pass rate (%) across training steps. Unlike math, counterfactual weighting provides no consistent benefit for code generation.

Step	CF	Vanilla	Δ
25	52.7	51.6	+1.1
100	52.4	51.9	+0.5
200	52.1	53.4	-1.3
250	51.3	54.2	-2.9
300	53.7	54.0	-0.3

unique path to that "answer sink," producing a strong counterfactual signal. In code, correctness requires multiple structural elements - control flow, variable bindings, return statements - to be simultaneously correct. No single span's removal catastrophically damages output probability in the same way. Additionally, our arithmetic-focused span detection (equations, calculations) fails to capture code-relevant structure such as conditionals, loops, and function calls.

This negative result clarifies the method's scope: counterfactual importance weighting is effective when task success depends on a small number of critical reasoning steps with a concentrated verification signal, as in mathematical problem-solving.

C Historical Context: Counterfactual Methods

Foundations and vision origins. The theoretical basis for counterfactual reasoning derives from Pearl's causal hierarchy (Pearl, 2009) and Rubin's potential outcomes framework (Rubin, 1974), which formalize the distinction between observational correlation and interventional causation. In deep learning, Zeiler & Fergus (2014) introduced **occlusion sensitivity**—systematically masking image regions and measuring prediction changes—establishing the "mask and measure" paradigm for importance estimation. This directly implements Pearl's do-operator: importance is revealed not by what correlates with predictions, but by what changes them when removed.

Extension to NLP. Li et al. (2016) adapted occlusion analysis to text through **representation erasure**, removing words or hidden dimensions and observing output changes. They further introduced reinforcement learning to find minimal erasure sets that flip predictions. Subsequent work formalized these intuitions: Integrated Gradients (Sundararajan et al., 2017) provided axiomatic foundations for attribution, while the "Attention is not Explanation" debate (Jain & Wallace, 2019) clarified that attention weights reveal where models look, but only intervention reveals what causally matters.

Causal analysis in transformers. Vig et al. (2020) introduced **causal mediation analysis** to transformers, distinguishing whether information *exists* in representations from whether it is *used*. Their intervention procedure isolates causal contributions via running counterfactual inputs while restoring specific components. Meng et al. (2022) extended this to **causal tracing**, using clean/corrupted/patched forward passes to locate where factual knowledge is stored, enabling targeted model editing. These methods demonstrated that causal understanding of transformers is both achievable and actionable.

Counterfactual credit assignment in RL. Counterfactual reasoning entered reinforcement learning through COMA (Foerster et al., 2018), which uses baselines marginalizing over alternative actions to isolate each agent's causal contribution. Mesnard et al. (2021) formalized counterfactual credit assignment by conditioning value functions on future outcomes to isolate each action's causal contribution to returns. For LLMs specifically, VinePPO (Kazemnejad et al., 2024) demonstrated that standard value networks fail at credit assignment in reasoning tasks, proposing Monte Carlo estimation from intermediate states.

D Additional Analysis Details

Span position matters. We partition spans into early (first third), middle, and late (last third) positions within each completion. Middle spans are most important (mean drop $= -270$), followed by late (-233) and early (-201). All pairwise differences are highly significant ($p < 10^{-10}$). This aligns with typical solution structure: early spans restate the problem, middle spans contain core derivations, and late spans state conclusions.

Span length correlates with importance. Longer spans receive higher importance scores ($r = -0.56$, $p < 10^{-100}$). This is expected: longer spans contain more reasoning content and their removal causes greater disruption. However, the moderate correlation indicates length alone does not determine importance - a verbose restatement can be long but unimportant.

Correct vs. incorrect completions. Importance distributions differ slightly between correct (N=2,353 spans, mean $= -221$) and incorrect (N=11,384 spans, mean $= -235$) completions, with incorrect completions showing marginally higher importance scores ($p = 0.012$). However, this small difference (6%) suggests our method primarily measures *structural* importance - which spans are causally connected to the answer - rather than *correctness*. A span can be critically important for producing a wrong answer.

Distractor spans. The 481 spans with positive importance drops (3.5%) represent content that *hurts* answer probability when present. Of these, 76.1% occur in incorrect completions, suggesting verbose or redundant reasoning correlates with failure. Manual inspection reveals distractors are typically step headers ("Step 1: Calculate the total miles..."), redundant restatements of given information ("Humans have 2 legs"), or verbose setup without computational content. The strongest distractor (drop $= +633$) is the span "Step 1: Calculate the total miles Jerome plans to ride in the first 12 days" - pure scaffolding that the model navigates better without.

Qualitative example. Table 8 shows importance scores for a GSM8K problem where Oliver avoids mango dishes at a buffet. The problem requires computing which dishes contain mango (via fractions and sums) then subtracting from the total.

The method clearly differentiates critical reasoning from scaffolding. The fraction calculation "$\frac{1}{6} \times 36 = 6$" receives highest importance ($|\Delta| = 806$) because it derives a quantity not given in the problem. The sum of mango dishes ($|\Delta| = 646$) aggregates this result. In contrast, restatements of given information ("Total dishes: 36", "Mango jelly: 1") receive 4–5$\times$ lower importance. Notably, the final arithmetic "$36 - 10 + 2 = 28$" scores lowest ($|\Delta| = 102$)—once the mango count is established, the subtraction is mechanical.

Table 8: Span importance for a GSM8K solution (Answer: 28). Problem: 36 dishes; 3 have mango salsa; $\frac{1}{6}$ have fresh mango; 1 has mango jelly. Oliver avoids mango but can pick it out of 2 fresh mango dishes. How many can he eat?

Step	Reasoning Span	$\lvert\Delta\rvert$
–	Break down the problem step by step	387
1	Total number of dishes: 36	172
2	Dishes with mango salsa: 3	211
3	Fresh mangoes: $\frac{1}{6} \times 36 = 6$ dishes	**806**
4	Dishes with mango jelly: 1	183
5	Total mango: $6+3+1 = 10$ dishes	**646**
6	Oliver can pick mango out of: 2	148
7	Final: $36 - 10 + 2 = 28$	102
–	Oliver can eat 28 dishes	190