PEERCREDIT: CROSS-PROMPT TOKEN CREDIT ASSIGNMENT FOR REINFORCEMENT LEARNING WITH VERIFIABLE REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) improves reasoning models using automatically checkable outcome rewards, but its response-level advantages do not specify how negative credit should be distributed across tokens in failed responses. Existing methods derive finer-grained signals mainly from rollouts of the same prompt, leaving cross-prompt reward information unused. We propose PeerCredit, which introduces cross-prompt reward information as an additional source of evidence for negative credit assignment. PeerCredit constructs reward-weighted peer gradients, measures token-level gradient alignment, removes variation explained by token probability, relative token position, and response length, and redistributes a fixed negative-credit budget so that tokens better aligned with reward-preferred peer gradients receive less penalty while the total budget is preserved. Across eight benchmarks spanning mathematical reasoning and tool use, PeerCredit achieves the best mathematical average at all three Qwen3 scales, with gains of up to **15.0** points over GRPO and **2.7** points over ResRL, while also leading on both tool-use benchmarks. Further analyses support cross-prompt reward evidence as a useful principle for improving fine-grained credit assignment in RLVR beyond within-prompt supervision.

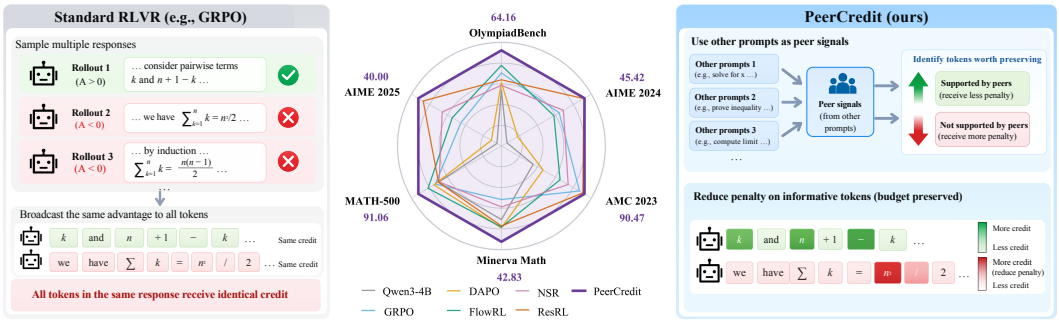


Figure 1: **Left**: GRPO assigns identical credit to all tokens in the same response. **Middle**: strongest Avg@16 results across six mathematical reasoning benchmarks with Qwen3-4B. **Right**: cross-prompt peer signals redistribute negative credit across tokens while keeping its total budget fixed.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become an effective way to improve language-model reasoning using automatically checkable outcomes (Shao et al., 2024). The supervision provided by RLVR, however, is much coarser than the updates it produces. A verifier evaluates an entire response, while policy optimization changes the probabilities of individual tokens. In GRPO, the same reward-derived advantage is assigned to every token in a response. As a result, when a response fails because of a small number of incorrect steps, useful intermediate reasoning can be penalized together with the actual errors. This raises a basic credit-assignment question: where should negative credit be placed within a failed response?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Recent methods have introduced token-dependent learning signals to refine response-level feedback (Lin et al., 2026; Chen et al., 2026; Zhang et al., 2026). Most of this evidence is obtained from the trajectories associated with the current prompt. We consider another source that is already present in an RLVR batch: reward information from other prompts. Although these prompts describe different problems, their updates act on the same model parameters. A reward-weighted direction estimated from one problem can therefore provide evidence about whether suppressing a token on another problem is locally compatible with reward-improving directions estimated elsewhere.

This view gives negative credit a direct optimization interpretation. Let G_i denote the score gradient of a negative-advantage target token and H_h a reward-improving direction estimated from a peer prompt. Their agreement $S_{ih} = \langle G_i, H_h \rangle_F$ measures whether increasing the target token's probability is locally compatible with the direction favored by the peer reward signal. When $S_{ih} > 0$, penalizing the token moves locally against that direction. Tokens with the same response-level negative advantage can therefore differ in the extent to which their suppression conflicts with reward-supported behavior on other problems. We refer to this mismatch as negative-credit misassignment. We therefore define this misassignment by the compatibility of a token-level update with reward-supported cross-prompt directions, rather than by semantic token correctness.

We propose PeerCredit to use this evidence for negative-credit assignment. PeerCredit constructs reward-weighted directions from peer prompts and measures their agreement with target-token score gradients. It removes variation in these scores that is explained by token probability, relative position, and response length, then aggregates the remaining signal into token-level peer support. This support determines how a fixed reference negative-credit budget is redistributed across eligible tokens. Tokens with stronger support receive less penalty, while the released credit is assigned elsewhere within the same target prompt. PeerCredit therefore reallocates negative credit across tokens while preserving the prompt-level reference budget. The resulting allocation is unique and guarantees nonnegative improvement in the residualized peer-support objective relative to the matched reference.

Experiments on six shared mathematical reasoning benchmarks show that PeerCredit improves mean Avg@16 over ResRL by 2.3, 2.7, and 0.6 points on Qwen3-1.7B/4B/8B, respectively, and by 15.0, 6.6, and 9.1 points over GRPO, with the best Qwen3-4B run exceeding ResRL by 5.3 points. Pass@k results further show that PeerCredit maintains leading performance from Pass@1 to Pass@128, covering both single-sample reliability and high-budget exploration. PeerCredit also achieves the best overall score among matched 8B post-trained methods on both BFCL and API-Bank. Ablations isolate the contributions of cross-prompt exchange, reward–trajectory correspondence, score residualization, and token-level localization. Under the same reference-budget rule, removing cross-prompt credit exchange lowers the seven-benchmark average by 6.93 points. Shuffling peer rewards or replacing token-level reassignment with response-level allocation also reduces performance, supporting the contribution of reward-conditioned cross-prompt information to token-wise credit reassignment.

Contributions. Our contributions are threefold:

- We identify negative-credit misassignment in RLVR: tokens with the same negative response advantage can differ in whether their suppression conflicts with reward-improving directions from other problems. A cross-prompt gradient analysis formalizes this mismatch and explains why negative-credit placement matters independently of its total magnitude.

- We introduce PeerCredit, a unified cross-prompt negative-credit reassignment method. PeerCredit turns reward differences from other prompts into token-level support and uses this evidence to redistribute a fixed reference negative-credit budget, so that peer information changes where negative credit is applied without changing its total reference amount.

- We validate PeerCredit across math and tool-use tasks. PeerCredit improves mean Avg@16 over ResRL by 2.3, 2.7, and 0.6 points on Qwen3-1.7B/4B/8B, respectively, with the best Qwen3-4B run exceeding ResRL by 5.3 points. It maintains leading Pass@k performance from 1 to 128 samples and leads matched 8B baselines on both BFCL and API-Bank. Removing cross-prompt credit exchange reduces the seven-benchmark average by 6.93 points, supporting the contribution of peer information to negative-credit reassignment.

2 Related Work

Reinforcement Learning with Verifiable Rewards. RLVR optimizes language models with verifiable outcome rewards and underlies recent reasoning systems (Shao et al., 2024; Guo et al., 2025a). PPO uses a learned critic, whereas RLOO and GRPO enable critic-free policy optimization with sampled baselines (Schulman et al., 2017; Ahmadian et al., 2024; Shao et al., 2024). DAPO refines clipping, sampling, and token-level loss aggregation; FlowRL matches reward-dependent target distributions; and NSR strengthens negative reinforcement (Yu et al., 2025; Zhu et al., 2026; 2025). PeerCredit instead retains response-level advantages and redistributes a fixed negative-credit budget across tokens.

Fine-Grained Credit Assignment. Fine-grained credit can come from process supervision or learned process rewards (Uesato et al., 2022; Lightman et al., 2024; Cui et al., 2026). VinePPO, SPO, and SSVPO estimate credit at token, segment, and step granularity (Kazemnejad et al., 2025; Guo et al., 2025b; Li et al., 2026); RED redistributes holistic rewards to tokens (Li et al., 2025); and SC-GRPO derives token weights from self-conditioned verified trajectories (Shan et al., 2026). Recent RLVR methods further exploit entropy, attention flow, representation residuals, hidden-state distances, and token-gradient structure (He et al., 2026; Mou et al., 2026; Dong et al., 2026; Lin et al., 2026; Chen et al., 2026; Zhang et al., 2026). Related gradient-based methods study example influence, reweighting, or update compatibility (Koh & Liang, 2017; Pruthi et al., 2020; Ren et al., 2018; Yu et al., 2020; Yang et al., 2026). PeerCredit differs by using reward-weighted directions from other prompt groups under fixed rewards, base advantages, and reference budgets.

3 Preliminaries

Consider an RLVR update with N prompts. For each prompt x_p, the old policy π_{θ_0} samples G responses $y_{pj} = (y_{pj1}, \ldots, y_{pjT_{pj}})$, each assigned a verifier reward R_{pj}. Following GRPO (Shao et al., 2024), the response-level advantage is computed from rewards within the same prompt group:

$$A_{pj} = \frac{R_{pj} - \bar{R}_p}{s_p + \epsilon_A}, \qquad \bar{R}_p = \frac{1}{G}\sum_{j=1}^{G} R_{pj}, \qquad A_{pjt} = A_{pj}, \tag{1}$$

where s_p is the within-group reward standard deviation. Thus, all valid tokens in a response inherit the same response-level advantage.

For a valid token $i = (p, j, t)$, let $\ell_i^{\text{base}}(\theta; A_i)$ denote the token-level loss used by the underlying policy optimizer. The corresponding minibatch objective can be written as

$$\mathcal{L}_{\mathcal{B}}^{\text{base}}(\theta) = \sum_{i \in \mathcal{B}} \alpha_i \, \ell_i^{\text{base}}(\theta; A_i), \tag{2}$$

where $\alpha_i > 0$ is the coefficient induced by token-level loss aggregation. PeerCredit retains the verifier reward, GRPO advantage, and base token-level objective ℓ_i^{base}. The exact clipped objective and aggregation coefficient are given in Appendix A.8.

We focus on negative credit assigned to unsuccessful responses. For prompt p, define the eligible negative-token set as

$$\mathcal{N}_p = \{i = (p, j, t) : R_{pj} \le 0,\ A_i < 0\}. \tag{3}$$

For nonempty $\mathcal{N}_p$ and each $i \in \mathcal{N}_p$, its base negative-credit magnitude is

$$q_i = \alpha_i |A_i|, \qquad Q_p = \sum_{i \in \mathcal{N}_p} q_i, \qquad m_i = \frac{q_i}{Q_p}, \qquad \sum_{i \in \mathcal{N}_p} m_i = 1. \tag{4}$$

Here q_i denotes the magnitude of token i's scalar negative-credit coefficient; Q_p is the total negative-credit mass of the target prompt, and m_i is token i's normalized share. These base quantities define the fixed prompt-level credit mass, while cross-prompt evidence determines its allocation over $\mathcal{N}_p$.

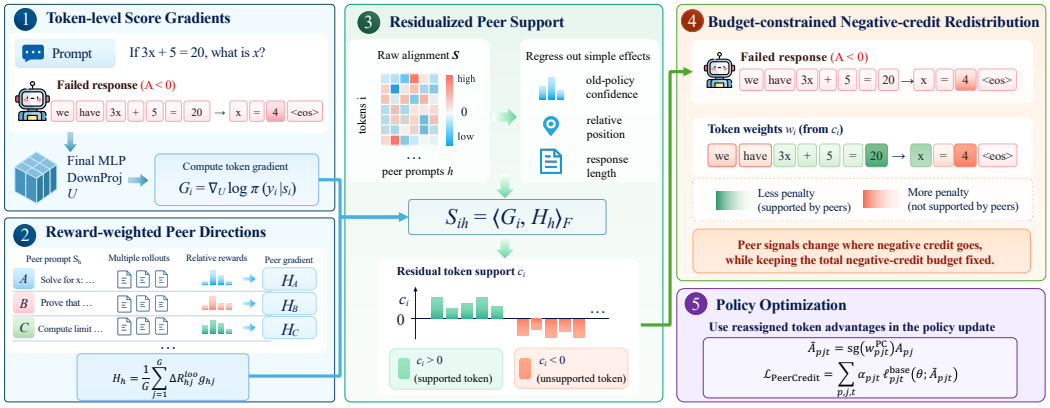


Figure 2: Overview of PeerCredit. **Modules 1–2**: compute target-token and reward-weighted peer gradients. **Module 3**: measure token–peer alignment, residualize probability, position, and length effects, and aggregate peer evidence. **Modules 4–5**: redistribute a fixed negative-credit budget into token-specific weights for the policy update.

4 PeerCredit

PeerCredit uses reward information from other prompts to determine where negative credit should be placed within a target prompt. Starting from the negative-credit pool $\mathcal{N}_p$ and masses m_i defined in Section 3, it computes target-token and peer gradients with respect to the same model parameter block, uses peer rewards to identify reward-preferred directions, and measures how strongly each target token agrees with these directions. The resulting cross-prompt evidence is then used to redistribute a fixed reference negative-credit budget across eligible tokens. Peer prompts determine the token-wise allocation under a reference budget computed independently of peer information. Figure 2 summarizes this computation from token score gradients to the final policy update.

4.1 Token-Level Score Gradients

To compare tokens from different prompts, their gradients must be defined with respect to the same model parameter block. Let d_{model} and d_{ff} denote the model dimension and feed-forward dimension, respectively. We choose the `down_proj` matrix of the MLP in the final Transformer block, $U \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$. This choice keeps the computation tractable while retaining the exact score gradient with respect to the selected parameter block: the linear projection admits an outer-product factorization, and its location in the final block requires differentiation only through the final RMSNorm and LM head rather than through additional Transformer blocks. It also avoids the vocabulary-sized dimension of the LM head.

At the position predicting token i, let $a_i \in \mathbb{R}^{d_{\text{ff}}}$ be the input to U, and let $z_i \in \mathbb{R}^{d_{\text{model}}}$ denote the residual representation immediately before the final RMSNorm. We write $z_i = h_i + U a_i$, where $h_i \in \mathbb{R}^{d_{\text{model}}}$ collects the residual terms independent of U; both a_i and h_i are fixed with respect to this parameter block. With s_i denoting the prefix before y_i,

$$e_i = \nabla_{z_i} \log \pi_{\theta_0}(y_i \mid s_i), \qquad G_i = \nabla_U \log \pi_{\theta_0}(y_i \mid s_i) = e_i a_i^\top, \\ \langle G_i, G_k \rangle_F = (e_i^\top e_k)(a_i^\top a_k), \qquad \forall\, i, k. \tag{5}$$

Here $e_i \in \mathbb{R}^{d_{\text{model}}}$ is the gradient of the token log-probability with respect to the final residual representation, whereas $G_i \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ is its score gradient with respect to U. The exact outer-product form allows token-gradient inner products to be computed without materializing a full $d_{\text{model}} \times d_{\text{ff}}$ matrix for every token. The derivation is given in Appendix A.1.

4.2 Reward-Weighted Peer Directions

To associate token-level parameter directions with reward, we use reward variation among rollouts of other prompts. For peer prompt x_h, let $g_{hj} = \sum_{t=1}^{T_{hj}} G_{hjt}$ denote the score gradient of rollout y_{hj}.

We define

$$\Delta R_{hj}^{\mathrm{loo}} = R_{hj} - \tfrac{1}{G-1}\sum\nolimits_{j' \neq j} R_{hj'}, \qquad H_h = \tfrac{1}{G}\sum\nolimits_{j=1}^{G} \Delta R_{hj}^{\mathrm{loo}} g_{hj}. \tag{6}$$

The reward contrast is positive for rollouts that outperform the other samples from the same prompt and negative for those that underperform. Accordingly, H_h combines response score gradients according to their relative rewards and summarizes the parameter direction favored by peer prompt h. Under independent on-policy sampling, H_h is an unbiased estimator of $\nabla_U \mathbb{E}[R \mid x_h]$; the derivation is given in Appendix A.2.

For target prompt p, let $\mathcal{P}_p$ denote the peer prompts whose canonical problem keys differ from that of the target. For $i \in \mathcal{N}_p$ and $h \in \mathcal{P}_p$, we measure token–peer agreement by

$$S_{ih} = \langle G_i, H_h \rangle_F = e_i^\top H_h a_i. \tag{7}$$

A positive S_{ih} means that the parameter change favored by peer h would locally increase the probability of target token i. Since i currently receives negative credit, a larger positive score provides evidence for reducing its penalty. Conversely, a negative score indicates that suppressing the token agrees with the peer reward signal. Thus S_{ih} quantifies reward-conditioned cross-prompt support for locally increasing the target token probability.

4.3 Residualized Peer Support

Raw token–peer agreement can vary with token confidence, position, and response length. We therefore remove the component explained by these simple token features before aggregating evidence across peers. For token $i = (p, j, t)$, we use $f_i = [1,\ 1 - p_i^{\mathrm{old}},\ (1 - p_i^{\mathrm{old}})^2,\ t/T_i,\ \log T_i]^\top$. Here $p_i^{\mathrm{old}} = \pi_{\theta_0}(y_i \mid s_i)$. The probability terms capture linear and quadratic dependence on token confidence, while t/T_i and $\log T_i$ account for relative position and response length.

Let F stack the feature vectors $f_i^\top$ over $i \in \mathcal{N}_p$, and let S contain the corresponding scores S_{ih} from Equation 7. Using the negative-credit shares m_i from Section 3, define $D_m = \mathrm{diag}(m)$, $\langle u, v \rangle_m = \sum_i m_i u_i v_i$, and $\|u\|_m^2 = \langle u, u \rangle_m$. With uniform peer weights $\omega_{ph} = |\mathcal{P}_p|^{-1}$, collected in ω_p, we compute

$$Z = S - F(F^\top D_m F)^\dagger F^\top D_m S, \quad c = \frac{Z\omega_p}{\sqrt{\sum_{h \in \mathcal{P}_p} \omega_{ph} \|Z_{:h}\|_m^2}}. \tag{8}$$

We set $c = 0$ when the denominator is numerically zero. The vector $Z_{:h}$ is the part of peer h's agreement scores not explained by the selected features, and c_i summarizes this residual evidence across peers for target token i. The pooled normalization also avoids amplifying a small mean score when different peers largely cancel one another.

By construction, $F^\top D_m c = 0$ and $\|c\|_m \leq 1$. Thus c removes weighted linear variation in the selected feature space and has a bounded scale before entering the credit allocator. The numerical residualization procedure and degenerate cases are described in Appendix B.2.

4.4 Budget-Constrained Negative-Credit Redistribution

The peer-support score c_i indicates where negative credit should be reduced, but changing token weights can also change the total strength of negative learning. We therefore fix a prompt-level reference budget before using peer information.

A probability-based rule computed from the old policy gives a scalar β_p, with the exact construction deferred to Appendix A.4. The matched control assigns the same reference weight $w_i = \beta_p$ to every token in $\mathcal{N}_p$. PeerCredit preserves this reference budget and uses c_i to redistribute it across tokens.

We optimize the token weights directly:

$$w^\star = \arg \min_{\substack{w_{\min} \leq w_i \leq w_{\max} \\ \sum_i m_i w_i = \beta_p}} \sum_{i \in \mathcal{N}_p} m_i \left[\tfrac{1}{2}(w_i - \beta_p)^2 + \rho c_i (w_i - \beta_p) \right]. \tag{9}$$

The first term keeps the weights close to the uniform reference β_p, while the second lowers the weight of tokens with larger peer-support scores. The equality constraint preserves the weighted mean weight, so reducing the penalty on some tokens must be compensated by increasing it elsewhere.

The objective is strictly convex and has the unique solution

$$w_i^\star = \mathrm{clip}(\beta_p - \rho c_i - \lambda_p,\, w_{\min}, w_{\max}), \qquad \sum_{i \in \mathcal{N}_p} q_i w_i^\star = \beta_p Q_p. \tag{10}$$

Here λ_p is a scalar chosen to satisfy the budget constraint. Hence a larger c_i produces no larger $w_i^\star$ and no greater negative credit, with the prompt-level reference budget preserved by construction. If $c = 0$, all eligible tokens recover the matched reference $w_i^\star = \beta_p$.

The allocation is unique and improves the residualized peer-support objective relative to the matched reference. The full statement and proof are given in Appendix A.6.

4.5 Policy Optimization

PeerCredit changes the token-wise credit used by the policy objective. For token $i = (p, j, t)$, we set $w_{pjt}^{\mathrm{PC}} = \alpha_+$ when $R_{pj} > 0$, $w_{pjt}^{\mathrm{PC}} = w_{pjt}^\star$ when $i \in \mathcal{N}_p$, and $w_{pjt}^{\mathrm{PC}} = 1$ otherwise, and define $\widetilde{A}_{pjt} = \mathrm{sg}(w_{pjt}^{\mathrm{PC}}) A_{pj}$. All weights are computed from the frozen outer batch and remain fixed during the subsequent policy update.

Let $r_{pjt}(\theta) = \pi_\theta(y_{pjt} \mid x_p, y_{pj,<t}) / \pi_{\theta_0}(y_{pjt} \mid x_p, y_{pj,<t})$, and define $\mathcal{C}_{pjt} \equiv \mathcal{C}_{pjt}(\theta) = \min\Big\{ r_{pjt}(\theta)\widetilde{A}_{pjt}, \mathrm{clip}(r_{pjt}(\theta), 1-\epsilon_-, 1+\epsilon_+)\widetilde{A}_{pjt} \Big\}$. Let $\mathbb{E}_{\mathcal{B}}$ denote expectation over $x_p \sim \mathcal{D}$ and $\{y_{pj}\}_{j=1}^G \sim \pi_{\theta_0}(\cdot \mid x_p)$. The complete PeerCredit policy objective is

$$\mathcal{J}_{\mathrm{PC}}(\theta) = \mathbb{E}_{\mathcal{B}} \left[\sum_{j=1}^{G} \sum_{t=1}^{T_{pj}} \alpha_{pjt} \left(\mathcal{C}_{pjt} + \mathbf{1}_{\{\widetilde{A}_{pjt} < 0\}} \left[\max\Big\{ c_{\mathrm{dual}} \widetilde{A}_{pjt}, \mathcal{C}_{pjt} \Big\} - \mathcal{C}_{pjt} \right] \right) \right]. \tag{11}$$

Here α_{pjt} is the coefficient induced by token-mean loss aggregation, and c_{dual} is the dual-clip coefficient for negative advantages. When $\widetilde{A}_{pjt} \geq 0$, the indicator term vanishes and Equation 11 reduces to the standard clipped surrogate. When $\widetilde{A}_{pjt} < 0$, the additional term implements the negative-side dual clip. PeerCredit modifies the base RL objective through $\widetilde{A}_{pjt}$, while retaining the verifier reward, GRPO advantage, importance ratio, clipping rule, and optimizer. Numerical values of α_+, ϵ_-, ϵ_+, and c_{dual} are given in Appendix C.

5 Experiments

5.1 Experimental Setup

We evaluate PeerCredit on six mathematical reasoning benchmarks: AIME 2024/2025 and AMC 2023 (Mathematical Association of America, 2026b;a), MATH-500 (Lightman et al., 2024), Minerva Math (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024), together with two tool-use benchmarks, BFCL v3 (Patil et al., 2025) and API-Bank (Li et al., 2023). Mathematical experiments use Qwen3-1.7B/4B/8B (Team, 2025) and compare the pretrained backbone with GRPO (Shao et al., 2024), DAPO (Yu et al., 2025), FlowRL (Zhu et al., 2026), NSR (Zhu et al., 2025), and ResRL (Lin et al., 2026). Tool-use experiments use Qwen3-8B.

For mathematics, we train on DAPO-Math-17k (Yu et al., 2025) for 200 steps with 256 prompts per batch and four rollouts per prompt. Qwen3 models run in no-thinking mode with 2,048/4,096 prompt/response limits; training uses temperature 1.0, top-$p = 1.0$, and top-$k = -1$. For tool use, we follow ToolRL (Qian et al., 2025) and train on `rlla_4k` with thinking enabled. Mathematics evaluation independently samples 16 responses per problem with temperature 0.6, top-$p = 0.95$, top-$k = 50$, and an 8,192-token generation limit, and reports Avg@16. Tool-use experiments follow their task-specific evaluation protocols. Complete settings are provided in Appendix C.

5.2 Main Results

Mathematical reasoning. Table 1 shows a consistent advantage across model scales. PeerCredit achieves the highest six-benchmark mean Avg@16 on Qwen3-1.7B/4B/8B, outperforming ResRL by

2.3, **2.7**, and **0.6** points, respectively, and GRPO by **15.0**, **6.6**, and **9.1** points. The best Qwen3-4B run further exceeds ResRL by **5.3** points. PeerCredit ranks first on all six benchmarks at 1.7B and retains the best overall average as the model scales to 4B and 8B.

Table 1: Mathematical reasoning results (Avg@16, %). Avg. is the mean over six benchmarks. PeerCredit values are means across available training seeds; per-seed results and sample standard deviations are reported in Appendix D.3. Best and second-best comparable results within each model size are bolded and underlined.

Model	Method	AIME24	AIME25	AMC23	MATH-500	Minerva	Olympiad	Avg.
Qwen3-1.7B	Backbone	11.0	9.8	43.9	69.5	26.1	38.1	33.1
	GRPO	12.3	13.8	54.2	71.5	27.5	36.0	35.9
	DAPO	10.0	8.4	57.5	70.9	30.3	33.6	35.2
	FlowRL	21.6	15.8	58.4	76.9	30.5	48.6	42.0
	NSR	27.0	20.4	66.7	83.5	33.9	53.5	47.5
	ResRL	26.9	21.3	66.9	84.4	35.5	56.6	48.6
	PeerCredit	**27.2**	**25.7**	**71.6**	**85.9**	**37.0**	**58.0**	**50.9**
Qwen3-4B	Backbone	20.0	17.3	56.9	77.8	36.9	48.2	42.9
	GRPO	37.1	27.7	87.2	79.9	31.5	55.1	53.1
	DAPO	23.5	18.9	63.4	80.8	39.1	51.2	46.2
	FlowRL	35.4	30.2	74.5	84.7	38.9	58.1	53.6
	NSR	38.5	33.1	79.8	77.4	33.5	50.1	52.1
	ResRL	**45.2**	38.6	**89.4**	77.8	38.6	52.3	57.0
	PeerCredit	43.7	**38.8**	88.1	**87.5**	**40.2**	**60.3**	**59.7**
Qwen3-8B	Backbone	25.4	18.1	61.4	77.6	39.2	48.6	45.1
	GRPO	36.3	29.2	78.0	89.4	42.1	62.0	56.2
	DAPO	24.2	24.0	71.3	76.2	35.3	43.6	45.8
	FlowRL	47.7	33.3	85.8	92.1	44.6	68.5	62.1
	NSR	**55.4**	38.5	**89.8**	87.3	40.0	60.6	61.9
	ResRL	50.8	41.1	89.7	92.7	**46.0**	68.1	64.7
	PeerCredit	53.3	**41.8**	89.0	**93.1**	45.0	**69.4**	**65.3**

Figures 3 and 4 provide a complementary view across inference budgets. Whereas Avg@16 averages correctness over sampled responses, Pass@k counts a problem as solved if at least one of k samples is correct. Pass@1 reflects single-sample reliability; higher-k Pass@k measures whether additional sampling uncovers a correct solution, assessing coverage across problems.

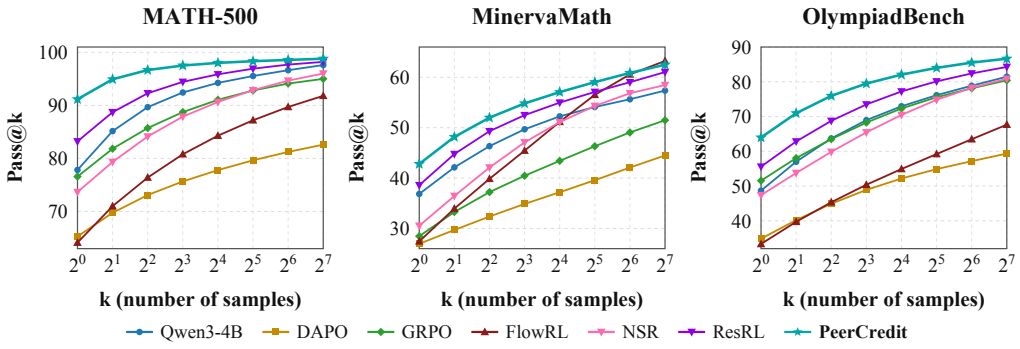


Figure 3: Pass@k performance on MATH-500, MinervaMath and OlympiadBench using Qwen3-4B.

PeerCredit maintains leading performance from Pass@1 to Pass@128, indicating that its Avg@16 gains improve low-budget reliability without sacrificing high-budget solution coverage.

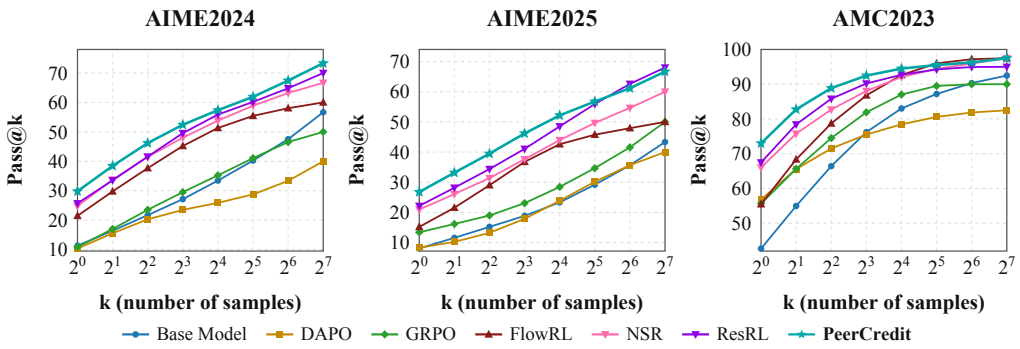


Figure 4: Pass@k performance on AIME 2024, AIME 2025, and AMC 2023 using Qwen3-1.7B.

Tool use. Tables 2 and 3 examine whether the same credit-assignment mechanism transfers from mathematical reasoning to structured tool interaction. On BFCL v3, PeerCredit-8B achieves an overall score of **69.06**, outperforming the matched ResRL-8B result of 68.95 while remaining competitive across both single-turn and multi-turn function-calling categories.

Table 2: Full BFCL v3 results. MT: Multi-Turn; ST: Single-Turn; MF: Missing Functions; MP: Missing Parameters; LC: Long Context. The PeerCredit row reports the mean across three seed/checkpoint runs, with individual results reported in Table 12. Bold/underline mark the best and second-best results among the listed post-training methods.

Method	MT Overall	Base	MF	MP	LC	ST Non-Live	ST Live	Overall
GPT-5-2025-08-07	28.50	33.50	29.50	23.00	28.00	72.92	58.25	52.65
Grok-4-0709	36.12	44.00	31.00	26.00	43.50	85.21	74.39	64.56
Qwen3-235B-A22B	40.12	49.00	41.00	29.50	41.00	87.90	77.03	67.69
ToolACE-2-8B	37.00	47.00	31.00	28.00	**42.00**	87.87	77.20	66.65
ResT-8B	<u>40.13</u>	**50.50**	<u>45.00</u>	<u>32.00</u>	33.00	**90.08**	79.03	68.76
NSR-8B	36.37	43.00	41.00	29.00	32.50	88.00	**80.23**	67.80
ResRL-8B	**41.25**	48.50	**47.00**	**34.00**	<u>35.50</u>	<u>89.46</u>	78.14	<u>68.95</u>
PeerCredit-8B	40.09	<u>50.00</u>	43.17	**34.00**	33.17	89.05	<u>79.64</u>	**69.06**

The API-Bank breakdown reveals a clearer gain as tool use becomes more compositional. PeerCredit improves overall accuracy from 66.50 with GRPO to **67.67** and from 64.15 with ResRL to **67.67**. Its Level 1 accuracy remains close to GRPO (74.94 vs. 75.44), while the gain grows to **6.61** points on Level 3 (47.07 vs. 40.46). Since API-Bank progresses from direct API calling to retrieval and planning with multiple API calls (Li et al., 2023), this pattern suggests that cross-prompt credit assignment is particularly useful when successful tool use depends on coordinating multiple decisions rather than executing a single known call.

Table 3: API-Bank results on Qwen3-8B. PeerCredit results are means across three training seeds; per-seed results and sample standard deviations are reported in Table 11. Bold/underline mark the best and second-best results among the listed post-training methods.

Method	Level 1 Accuracy	Level 2 Accuracy	Level 3 Accuracy	Overall Accuracy
SFT	69.17	56.72	36.64	60.64
TSFT	66.42	50.75	<u>44.27</u>	59.80
RSFT	67.42	53.73	42.75	60.47
GRPO	**75.44**	<u>64.18</u>	40.46	<u>66.50</u>
ResRL	72.68	56.72	41.98	64.15
PeerCredit	<u>74.94</u>	**64.68**	**47.07**	**67.67**

5.3 Ablation and Mechanistic Analysis

We examine which parts of PeerCredit account for its gains. All ablations use Qwen3-4B with seed 1001, and the same training protocol, with Gaokao 2023 included as an extra benchmark. The full method uses $\rho = 0.2$ and all peers ($M = \text{All}$). Detailed intervention definitions are provided in Appendix D.

Table 4: Ablation studies on Qwen3-4B across seven mathematical benchmarks (Avg@16, %).

Method	AIME24	AIME25	AMC23	MATH-500	Minerva	Olympiad	Gaokao23	Avg.
PeerCredit	45.42	40.00	**90.47**	**91.06**	**42.83**	**64.16**	**78.17**	**64.59**
w/o cross-prompt exchange	44.38	37.08	86.72	78.05	33.41	54.88	69.09	57.66
w/o score residualization	46.46	36.25	80.16	79.55	33.02	54.51	63.00	56.13
Shuffled peer rewards	48.33	40.42	86.72	77.34	35.00	52.10	67.61	58.22
Response-level allocation	**51.25**	**46.04**	88.44	74.19	29.76	48.00	67.03	57.81

Reward-conditioned cross-prompt information matters. Removing cross-prompt exchange reduces the average from 64.59 to 57.66 (-6.93), while shuffling peer rewards reduces it to 58.22 (-6.37). The latter preserves the peer trajectories and reward multiset but breaks their correspondence. These controls support reward conditioning, rather than cross-prompt exchange alone, as an important contributor to the gain.

Residualization and token localization contribute separately. Removing score residualization gives the largest drop, from 64.59 to 56.13 (-8.46), showing that raw alignment scores are substantially less useful for allocation. Replacing token-level weights with response-level weights reduces the average to 57.81 (-6.78) while preserving each response's negative-credit amount, supporting the value of within-response token localization. Figure 5 shows persistent credit redistribution throughout training while preserving the budget constraint numerically.

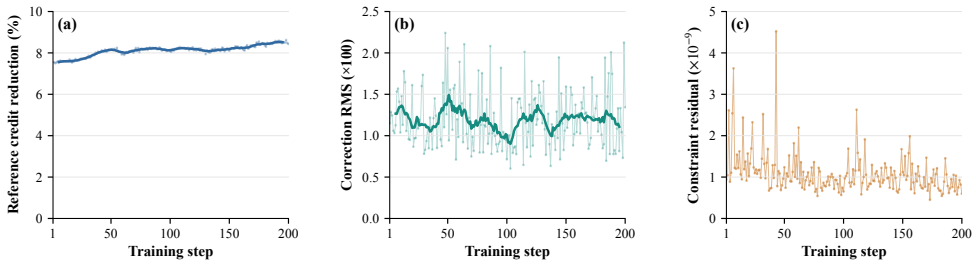


Figure 5: PeerCredit allocation diagnostics on Qwen3-4B: reference credit reduction and correction remain nonzero throughout training, while the budget residual stays near numerical precision.

6 Conclusion

We present PeerCredit, a cross-prompt credit-assignment method for RLVR that uses reward information from other problems to identify which tokens in a failed response should receive less negative credit. PeerCredit measures token–peer agreement in parameter-gradient space, removes variation explained by simple token statistics, and redistributes a fixed negative-credit budget. The resulting allocation is unique, preserves the reference budget, and guarantees nonnegative improvement in the residualized peer-agreement objective. Across six mathematical benchmarks, PeerCredit achieves the highest average at all three Qwen3 scales, improving mean Avg@16 over ResRL by 2.3, 2.7, and 0.6 points on 1.7B/4B/8B. It also maintains leading Pass@k performance from Pass@1 to Pass@128 and obtains the highest reported overall scores among the compared 8B post-training methods on BFCL and API-Bank. Together, these results support cross-prompt reward information as a useful signal for finer-grained credit assignment in RLVR.

REPRODUCIBILITY STATEMENT

Appendices A–D provide the complete method derivations, proofs, training and evaluation configurations, seed-level results, ablation definitions, and numerical implementation details required to reproduce the reported experiments.

AI USE STATEMENT

Generative AI assisted with literature/source checks, and manuscript/LaTeX editing. No empirical benchmark measurements were generated or inferred using generative AI. The authors are responsible for verifying all AI-assisted text, proofs, protocols, and artifacts before submission.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE-style optimization for learning from human feedback in LLMs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.662.

Xinzhu Chen, Wei He, Huichuan Fan, Wenzhe Niu, Zhongxiang Sun, Xuanru Wang, Jiuchong Gao, Jinghua Hao, Renqing He, and Weijie Yu. Hidden states know where reasoning diverges: Credit assignment via span-level wasserstein distance, 2026. URL https://arxiv.org/abs/2604.23318.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=9SkkifLopZ.

Zhichen Dong, Yang Li, Yuhan Sun, Weixun Wang, Yijia Luo, Zinian Peng, Wenbo Su, Cheng Yu, Bo Zheng, and Junchi Yan. How does reasoning flow? tracing attention-induced information flow for targeted RL in LLMs. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=Ym4KYa1n76.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025a. doi: 10.1038/s41586-025-09422-z.

Yiran Guo, Lijie Xu, Jie Liu, Ye Dan, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=9osvTOYbT4.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Yuhang He, Haodong Wu, Siyi Liu, Hongyu Ge, Hange Zhou, Keyi Wu, Zhuo Zheng, Qihong Lin, Zixin Zhong, and Yongqi Zhang. Where hindsight credit can reside: A signed-capacity view of token updates in rlvr, 2026. URL https://arxiv.org/abs/2604.11056.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Myx2kJFzAn.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894. PMLR, 2017.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-0278.

Jiahui Li, Lin Li, Tai-Wei Chang, Kun Kuang, Long Chen, Jun Zhou, and Cheng Yang. RED: Unleashing token-level rewards from holistic feedback via reward redistribution. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 4993–5022, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.252. URL https://aclanthology.org/2025.emnlp-main.252/.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. API-bank: A comprehensive benchmark for tool-augmented LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Yugu Li, Zehong Cao, Jianglin Qiao, and Siyi Hu. SSVPO: Effective step-level credit assignment for RL training of language models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=g33DGvnHYd.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Zihan Lin, Xiaohan Wang, Jie Cao, Jiajun Chai, Li Wang, Xiaodong Lu, Wei Lin, Ran He, and Guojun Yin. ResRL: Boosting LLM reasoning via negative sample projection residual reinforcement learning. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=kmN9ozKtGh.

Mathematical Association of America. MAA invitational competitions. https://maa.org/maa-invitational-competitions/, 2026a. Accessed September 2026.

Mathematical Association of America. American mathematics competitions. https://maa.org/student-programs/amc/, 2026b. Accessed September 2026.

Chaoli Mou, Zhan Zhuang, Xinning Chen, and Yu Zhang. Beyond uniform credit assignment: Selective eligibility traces for rlvr, 2026. URL https://arxiv.org/abs/2605.05965.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19920–19930, 2020.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. ToolRL: Reward is all tool learning needs. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3524.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4334–4343. PMLR, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Yingyu Shan, Yuhang Guo, Zihao Cheng, Zeming Liu, Xiangrong Zhu, Xinyi Wang, Jiashu Yao, Wei Lin, Hongru Wang, and Heyan Huang. Learning from own solutions: Self-conditioned credit assignment for reinforcement learning with verifiable rewards, 2026. URL https://arxiv.org/abs/2606.18810.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Qwen Team. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025. doi: 10.48550/ARXIV.2505.09388. URL https://doi.org/10.48550/arXiv.2505.09388.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback, 2022. URL https://arxiv.org/abs/2211.14275.

Ningyuan Yang, Weihua Du, Weiwei Sun, Sean Welleck, and Yiming Yang. GradAlign: Gradient-aligned data selection for LLM reinforcement learning. In *The Third Conference on Language Modeling*, 2026.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, YuYue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, LingJun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Ru Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Yonghui Wu, and Mingxuan Wang. DAPO: An open-source LLM reinforcement learning system at scale. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=2a36EMSSTp.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5824–5836, 2020.

Kaiyi Zhang, Wei Wu, and Yankai Lin. Delta: Discriminative token credit assignment for reinforcement learning from verifiable rewards, 2026. URL https://arxiv.org/abs/2605.21467.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=ftVlLG9cks.

Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Hengli Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, Qizheng Zhang, Lin Chen, Fanghao Shao, Bo Xue, Yunchong Song, Zhenjie Yang, Ganqu Cui, Ning Ding, Jianfeng Gao, Xiaodong Liu, Bowen Zhou, Hongyuan Mei, and Zhouhan Lin. FlowRL: Matching reward distributions for LLM reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lObnTKbm9U.

A Additional Method Details and Proofs

A.1 Exact Gradient of the Final MLP Down-Projection

We derive the factorization in Equation 5 and motivate the parameter-block choice used by PeerCredit. Let d_{model} and d_{ff} denote the model dimension and feed-forward dimension, respectively, and let $U \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$ be the `down_proj` matrix of the MLP in the final Transformer block. We use this block because the final MLP is position-wise and is followed by no further token-mixing Transformer block. Hence the log-probability of a target token depends on this parameter block only through the same-position term Ua_i, which yields the exact outer-product factorization below. Its location also lets us reconstruct the downstream derivative through only the final RMSNorm and LM head. The factorized representation stores only e_i and a_i and therefore introduces no vocabulary-sized stored dimension, although computing e_i still requires differentiation through the final RMSNorm and LM head. The resulting score gradient is exact with respect to the selected parameter block, which provides the common parameter space used for all cross-prompt comparisons.

At the position predicting token i, let $a_i \in \mathbb{R}^{d_{\text{ff}}}$ denote the input activation to U, and let $z_i \in \mathbb{R}^{d_{\text{model}}}$ denote the residual stream immediately before the final RMSNorm. We write

$$z_i = h_i + Ua_i, \tag{12}$$

where $h_i \in \mathbb{R}^{d_{\text{model}}}$ collects the terms independent of this parameter block. For a fixed sampled prefix, the selected down-projection U occurs only after a_i has been computed at this position; hence both a_i and h_i are independent of U. Define

$$\ell_i = \log \pi_{\theta_0}(y_i \mid s_i), \qquad e_i = \nabla_{z_i} \ell_i. \tag{13}$$

For an infinitesimal perturbation $\mathrm{d}U$, $\mathrm{d}z_i = (\mathrm{d}U)a_i$, and therefore

$$\mathrm{d}\ell_i = e_i^\top \mathrm{d}z_i = e_i^\top (\mathrm{d}U) a_i = \left\langle e_i a_i^\top, \mathrm{d}U \right\rangle_F. \tag{14}$$

Hence

$$G_i = \nabla_U \ell_i = e_i a_i^\top. \tag{15}$$

For any two tokens i and k,

$$\langle G_i, G_k \rangle_F = (e_i^\top e_k)(a_i^\top a_k). \tag{16}$$

Thus the factorization is exact for the selected parameter block, and token-gradient inner products can be computed from the factors e_i and a_i without materializing a $d_{\text{model}} \times d_{\text{ff}}$ matrix for each token.

A.2 Peer Estimator

For peer prompt x_h, let

$$\Delta R_{hj}^{\text{loo}} = R_{hj} - \frac{1}{G-1} \sum_{j' \neq j} R_{hj'} \tag{17}$$

be the leave-one-out reward contrast, and let

$$g_{hj} = \sum_{t=1}^{T_{hj}} G_{hjt} = \nabla_U \log \pi_{\theta_0}(y_{hj} \mid x_h) \tag{18}$$

be the response score gradient with respect to U. The sum is over valid sampled response tokens; padding is excluded, and a sampled end-of-sequence token is included when present. PeerCredit constructs

$$H_h = \frac{1}{G} \sum_{j=1}^{G} \Delta R_{hj}^{\text{loo}} g_{hj}. \tag{19}$$

The peer direction is constructed directly from response score gradients and leave-one-out reward contrasts, independently of target-side credit coefficients.

Unbiasedness. Assume that the G responses are independent samples from $\pi_{\theta_0}(\cdot \mid x_h)$ and that the reward function is fixed with respect to U. By the score-function identity, $\mathbb{E}[g_{hj} \mid x_h] = 0$. For $j' \neq j$, the leave-one-out baseline is independent of rollout j, so

$$\mathbb{E}[R_{hj'} g_{hj} \mid x_h] = \mathbb{E}[R_{hj'} \mid x_h]\,\mathbb{E}[g_{hj} \mid x_h] = 0. \tag{20}$$

Consequently,

$$\mathbb{E}[H_h \mid x_h] = \frac{1}{G}\sum_{j=1}^{G} \mathbb{E}[R_{hj} g_{hj} \mid x_h] \tag{21}$$

$$= \nabla_U \mathbb{E}_{y \sim \pi_{\theta_0}(\cdot \mid x_h)}[R(x_h, y)]. \tag{22}$$

This establishes the expected-reward interpretation used in the main text.

First-order interpretation of token–peer agreement. For a target token i, hold all other model parameters fixed and consider an infinitesimal update to the selected block, $U \leftarrow U + \eta H_h$. Taylor expansion gives

$$\log \pi_{U+\eta H_h}(y_i \mid s_i) - \log \pi_U(y_i \mid s_i) = \eta \left\langle \nabla_U \log \pi_U(y_i \mid s_i), H_h \right\rangle_F + O(\eta^2) \tag{23}$$

$$= \eta S_{ih} + O(\eta^2). \tag{24}$$

Thus $S_{ih} > 0$ means that the peer direction locally increases the target token log-probability.

Peer exclusion and weighting. Prompts with the same canonicalized text key are excluded from one another's peer panels. We construct canonical keys using Unicode NFKC normalization followed by whitespace normalization, so peer exclusion operates at the normalized-text level. For target prompt p,

$$\mathcal{P}_p = \{h : \mathrm{key}(x_h) \neq \mathrm{key}(x_p)\}, \qquad \omega_{ph} = |\mathcal{P}_p|^{-1}. \tag{25}$$

Uniform peer weighting includes peer groups with $H_h = 0$. If $\mathcal{P}_p = \varnothing$, PeerCredit returns the reference allocation.

A.3 Weighted Residualization and Pooled Normalization

For a fixed target prompt p, let $n_p = |\mathcal{N}_p|$ and $D_m = \mathrm{diag}(m_1, \ldots, m_{n_p})$. For token $i = (p, j, t)$, the feature vector used in the method is

$$f_i = \left[1,\ 1 - p_i^{\mathrm{old}},\ (1 - p_i^{\mathrm{old}})^2,\ t/T_i,\ \log T_i\right]^\top. \tag{26}$$

Stacking $f_i^\top$ into $F \in \mathbb{R}^{n_p \times 5}$ and the raw token–peer scores into S, the weighted least-squares coefficient and residual are

$$B^\star = (F^\top D_m F)^\dagger F^\top D_m S, \qquad Z = S - F B^\star. \tag{27}$$

The residual satisfies

$$F^\top D_m Z = 0. \tag{28}$$

Equivalently, let Q contain a basis for $\mathrm{col}(F)$ that is orthonormal under D_m, so that $Q^\top D_m Q = I$. The corresponding weighted projector is $P_F = QQ^\top D_m$, and $Z = (I - P_F)S$. In exact arithmetic, the same representation applies to rank-deficient F when Q spans $\mathrm{col}(F)$. The numerical implementation instead uses the retained feature subspace defined in Appendix B.2.

Let

$$\bar{z} = \sum_{h \in \mathcal{P}_p} \omega_{ph} Z_{:h}, \qquad E_p = \sum_{h \in \mathcal{P}_p} \omega_{ph} \|Z_{:h}\|_m^2. \tag{29}$$

When $E_p > 0$, PeerCredit uses

$$c = \frac{\bar{z}}{\sqrt{E_p}}, \tag{30}$$

and sets $c = 0$ otherwise. Equation 28 implies $F^\top D_m c = 0$. Moreover, convexity of the squared norm gives

$$\|c\|_m^2 = \frac{\|\sum_h \omega_{ph} Z_{:h}\|_m^2}{E_p} \leq \frac{\sum_h \omega_{ph} \|Z_{:h}\|_m^2}{E_p} = 1. \tag{31}$$

The pooled-energy denominator preserves the attenuation induced by disagreement among peer residuals, so cancellation across peers reduces the magnitude of the aggregate support score.

A.4 Probability-Derived Reference Budget

The probability-based reference determines the prompt-level negative-credit budget, while PeerCredit determines its token-wise allocation.

For eligible token i, define

$$u_i = 1 - p_i^{\text{old}}, \qquad \bar{u} = \sum_i m_i u_i, \qquad \sigma_u = \sqrt{\sum_i m_i (u_i - \bar{u})^2}. \tag{32}$$

If σ_u is numerically zero, set $v_i = 1$ for every eligible token. Otherwise define

$$\zeta_i = \frac{u_i - \bar{u}}{\sigma_u}, \qquad \sum_i m_i \zeta_i = 0, \qquad \sum_i m_i \zeta_i^2 = 1. \tag{33}$$

For $\kappa \geq 0$ we define

$$v_i(\kappa) = \text{clip}\left(1 + \kappa(\zeta_i - \nu_\kappa), 0.5, 1.5\right), \tag{34}$$

where ν_κ is chosen so that $\sum_i m_i v_i(\kappa) = 1$. A second scalar search chooses κ so that $\|v(\kappa) - \mathbf{1}\|_m = r_0$ whenever this radius is attainable under the box constraints; otherwise the limiting feasible allocation is used. Hence

$$\|v - \mathbf{1}\|_m \leq r_0. \tag{35}$$

PeerCredit uses the capped weighted mean of v as the uniform prompt-level reference:

$$\beta_p = \sum_i m_i \min\{v_i, 1\}, \qquad b_i = \beta_p. \tag{36}$$

Thus the reference used by both PeerCredit and the matched control is uniform within the eligible pool.

Range of β_p. Let $x_i = v_i - 1$. Since $\sum_i m_i x_i = 0$, the positive and negative absolute masses are equal, and

$$1 - \beta_p = \frac{1}{2} \sum_i m_i |x_i| \leq \frac{1}{2} \sqrt{\sum_i m_i x_i^2} = \frac{1}{2} \|v - \mathbf{1}\|_m \leq \frac{r_0}{2}. \tag{37}$$

For the value $r_0 = 0.2$, this gives $0.9 \leq \beta_p \leq 1$.

A.5 Normalized and Absolute Budget Forms

From Equation 4, $q_i = \alpha_i |A_i|$, $Q_p = \sum_{i \in \mathcal{N}_p} q_i$, and $m_i = q_i / Q_p$. Therefore the normalized budget constraint $\sum_i m_i w_i = \beta_p$ is exactly equivalent to

$$\sum_{i \in \mathcal{N}_p} q_i w_i = \beta_p Q_p. \tag{38}$$

Thus Equation 38 preserves the total scalar negative-credit budget assigned to the eligible pool.

A.6 Allocation Solution and Proof

The main text optimizes the token weights w directly. For a nonempty eligible pool, define the feasible set

$$\mathcal{W}_p = \left\{ w : w_{\min} \leq w_i \leq w_{\max},\ \sum_i m_i w_i = \beta_p \right\}. \tag{39}$$

Assuming $\beta_p \in [w_{\min}, w_{\max}]$, the uniform reference $\beta_p \mathbf{1}$ is feasible.

Theorem 1 (Budget preservation and residualized-support improvement)**.** *For any nonempty $\mathcal{N}_p$, $\rho > 0$, and $\beta_p \in [w_{\min}, w_{\max}]$, the solution of Equation 9 is unique and satisfies*

$$w_{\min} \leq w_i^\star \leq w_{\max}, \qquad \sum_i m_i w_i^\star = \beta_p, \qquad \sum_i q_i w_i^\star = \beta_p Q_p. \tag{40}$$

Let $\delta^\star = w^\star - \beta_p \mathbf{1}$ *and* $\Gamma_p = -\langle c, \delta^\star \rangle_m$. *Then*

$$\Gamma_p \geq \frac{\|\delta^\star\|_m^2}{\rho} \geq 0, \qquad \|\delta^\star\|_m \leq \rho \|c\|_m \leq \rho. \tag{41}$$

If $c = 0$, *then* $w^\star = \beta_p \mathbf{1}$.

Existence and uniqueness. The objective in Equation 9 is

$$f(w) = \sum_i m_i \left[\frac{1}{2}(w_i - \beta_p)^2 + \rho c_i (w_i - \beta_p) \right]. \tag{42}$$

Every $m_i > 0$, so f is strictly convex. The feasible set $\mathcal{W}_p$ is nonempty, closed, bounded, and convex. The minimizer therefore exists and is unique.

KKT characterization. Introduce a scalar multiplier λ_p for $\sum_i m_i w_i = \beta_p$. Ignoring the box constraints, stationarity gives $w_i - \beta_p + \rho c_i + \lambda_p = 0$. Projection onto $[w_{\min}, w_{\max}]$ gives

$$w_i^\star = \mathrm{clip}\left(\beta_p - \rho c_i - \lambda_p,\, w_{\min}, w_{\max}\right), \tag{43}$$

where λ_p is chosen so that $\sum_i m_i w_i^\star = \beta_p$.

Residualized-support improvement. Set $\delta = w - \beta_p \mathbf{1}$. The feasible perturbations satisfy $\langle \mathbf{1}, \delta \rangle_m = 0$ and the corresponding shifted box constraints, while the objective becomes $\frac{1}{2}\|\delta\|_m^2 + \rho \langle c, \delta \rangle_m$. The first-order optimality condition at $\delta^\star$ is

$$\langle \delta^\star + \rho c,\, \delta - \delta^\star \rangle_m \geq 0 \qquad \forall\, \delta \text{ feasible}. \tag{44}$$

The zero perturbation is feasible, so choosing $\delta = 0$ yields

$$-\|\delta^\star\|_m^2 - \rho \langle c, \delta^\star \rangle_m \geq 0. \tag{45}$$

Hence, with $\Gamma_p = -\langle c, \delta^\star \rangle_m$,

$$\Gamma_p \geq \frac{\|\delta^\star\|_m^2}{\rho} \geq 0. \tag{46}$$

Cauchy–Schwarz also gives $\Gamma_p \leq \|c\|_m \|\delta^\star\|_m$, which together with Equation 46 implies $\|\delta^\star\|_m \leq \rho \|c\|_m \leq \rho$. If $c = 0$, strict convexity makes the uniform reference $w^\star = \beta_p \mathbf{1}$ the unique solution.

A.7 Relation to the Raw Peer Direction

The guarantee in Theorem 1 concerns the residualized score c used by the allocator. We next relate the residualized guarantee to the raw peer direction.

Define

$$\bar{H}_p = \sum_{h \in \mathcal{P}_p} \omega_{ph} H_h, \qquad \bar{s} = S \omega_p, \qquad \bar{s}_i = \langle G_i, \bar{H}_p \rangle_F. \tag{47}$$

Let $s_F = P_F \bar{s}$ denote the component explained by the selected feature space. Since $(I - P_F)\bar{s} = Z\omega_p = \sqrt{E_p}\, c$,

$$\bar{s} = \sqrt{E_p}\, c + s_F. \tag{48}$$

At $\theta = \theta_0$, where the likelihood ratio equals one and the clipping terms are inactive, the eligible negative-pool contribution to the policy ascent direction on the selected parameter block is

$$D_p^{\mathrm{PC}} = -Q_p \sum_i m_i w_i^\star G_i, \qquad D_p^{\mathrm{Ctrl}} = -Q_p \sum_i m_i \beta_p G_i. \tag{49}$$

Their difference is

$$\Delta D_p = -Q_p \sum_i m_i \delta_i^\star G_i, \tag{50}$$

and therefore

$$\langle \bar{H}_p, \Delta D_p \rangle_F = -Q_p \langle \delta^\star, \bar{s} \rangle_m \tag{51}$$

$$= Q_p \left(\sqrt{E_p}\, \Gamma_p - \langle s_F, \delta^\star \rangle_m \right). \tag{52}$$

PeerCredit preserves the weighted negative-credit budget, equivalently the intercept moment. Equation 52 decomposes the raw peer-direction change into the residualized-support term and the feature-space term $\langle s_F, \delta^\star \rangle_m$. Theorem 1 guarantees nonnegative improvement in the residualized-support term.

When the unconstrained solution $\delta^\star = -\rho c$ lies inside the box constraints, c is orthogonal to the feature span and $\langle s_F, \delta^\star \rangle_m = 0$. In that case,

$$\langle \bar{H}_p, \Delta D_p \rangle_F = Q_p \sqrt{E_p}\, \Gamma_p \geq 0. \tag{53}$$

A.8 Loss Coefficients and Policy Objective

The main text indexes only valid response tokens, so padding masks do not appear in its mathematical definition. Let $\mathcal{H}(i)$ denote the local PPO microbatch containing token i, let $n_{\mathcal{H}(i)}$ be the number of valid response tokens in that microbatch, and let K_{acc} be the number of equal-sized microbatches accumulated into one PPO minibatch. Up to the common data-parallel averaging factor,

$$\alpha_i = \frac{1}{K_{\mathrm{acc}}\, n_{\mathcal{H}(i)}}. \tag{54}$$

Thus $q_i = \alpha_i |A_i|$ is the magnitude of token i's negative scalar coefficient in the base policy objective.

Final weighting rule. Using the notation of the main text,

$$w_i^{\mathrm{PC}} = \begin{cases} \alpha_+, & R_{pj} > 0, \\ w_i^\star, & i \in \mathcal{N}_p, \\ 1, & \text{otherwise}, \end{cases} \qquad \widetilde{A}_i = \mathrm{sg}(w_i^{\mathrm{PC}}) A_i. \tag{55}$$

For the configuration, $\alpha_+ = 0.1$, $w_{\min} = 0.1$, and $w_{\max} = 1$. In the binary-verifier setting, the final branch includes zero-advantage failed groups and therefore contributes no policy-gradient update; more generally, tokens outside the two modified sets retain the base weight.

Exact clipped objective. Let

$$r_i(\theta) = \frac{\pi_\theta(y_i \mid s_i)}{\pi_{\theta_0}(y_i \mid s_i)}, \qquad \bar{r}_i(\theta) = \mathrm{clip}\left(r_i(\theta), 1 - \epsilon_-, 1 + \epsilon_+\right), \tag{56}$$

and define the standard clipped surrogate

$$\mathcal{C}_i(\theta) = \min\left\{ r_i(\theta)\widetilde{A}_i,\, \bar{r}_i(\theta)\widetilde{A}_i \right\}. \tag{57}$$

Up to the common data-parallel averaging factor, which is applied equally to all compared methods, the minibatch maximization objective is

$$\mathcal{J}_{\mathcal{B}}^{\mathrm{PC}}(\theta) = \sum_{i \in \mathcal{B}} \alpha_i \left(\mathcal{C}_i(\theta) + \mathbf{1}_{\{\widetilde{A}_i < 0\}} \left[\max\left\{ c_{\mathrm{dual}} \widetilde{A}_i,\, \mathcal{C}_i(\theta) \right\} - \mathcal{C}_i(\theta) \right] \right). \tag{58}$$

Training minimizes the corresponding loss

$$\mathcal{L}_{\mathcal{B}}^{\mathrm{PC}}(\theta) = -\mathcal{J}_{\mathcal{B}}^{\mathrm{PC}}(\theta). \tag{59}$$

For $\widetilde{A}_i \geq 0$, the indicator vanishes and the objective is the standard clipped surrogate. For $\widetilde{A}_i < 0$, it reduces to $\max\{c_{\mathrm{dual}} \widetilde{A}_i, \mathcal{C}_i(\theta)\}$, which gives the negative-side dual-clip objective used in our experiments.

For the configuration, $\epsilon_- = \epsilon_+ = 0.2$ and $c_{\mathrm{dual}} = 3$. Entropy and actor-KL loss coefficients are zero. All peer directions, residualized scores, reference budgets, and token weights are computed from the frozen rollout batch before the PPO minibatch updates, and the returned weights remain detached throughout the policy update.

B Implementation and Numerical Details

B.1 Frozen Outer-Batch Computation

Algorithm 1 summarizes the PeerCredit weighting procedure. Rewards, base advantages, old-policy log-probabilities, prompt-group identities, and canonical problem keys are fixed before token weights are constructed. Prompt-group identities define target credit pools, while canonical keys define the eligible peer groups.

For the matched control, setting $c = 0$ yields the uniform reference allocation. Equation 9 then has the unique solution $w_i = \beta_p$ for every eligible negative token.

B.2 Feature Basis and Degenerate Panels

All pool-level linear algebra is evaluated in float64. Each feature column $F_{:k}$ is rescaled by

$$s_k = \left(\sum_i m_i F_{ik}^2 \right)^{1/2}, \tag{60}$$

using one when the scale is zero. After rescaling the feature columns, we compute the thin SVD

$$D_m^{1/2} F \ \mathrm{diag}(s_k^{-1}) = U_F \Sigma V^\top, \qquad Q = D_m^{-1/2} (U_F)_{\mathrm{keep}}. \tag{61}$$

The singular-value threshold is $10 \max(n, 5) \epsilon_{64} \sigma_{\max}$. Thus Q defines the weighted feature subspace after numerical rank truncation.

With k peer-score columns, define

$$E_{\mathrm{raw}} = \sum_{ih} m_i \omega_{ph} S_{ih}^2. \tag{62}$$

We set $c = 0$ whenever

$$E_p \le \left[100 \max(n, k) \epsilon_{64} \right]^2 \max\left(E_{\mathrm{raw}}, \mathrm{tiny}_{64} \right). \tag{63}$$

For an empty peer panel, we set $c = 0$. These conventions cover constant probabilities, rank-deficient feature matrices, and peer scores fully explained by the residualization features.

B.3 Dual Solution and Numerical Tolerances

We solve for the scalar multiplier λ_p in Equation 43 numerically. We use BFGS with gradient tolerance 10^{-12} and at most 1,000 iterations. Solutions with a dual-gradient residual above 10^{-10} are further refined by least squares with tolerance 10^{-13} and at most 1,000 function evaluations. After casting the weights to float32, budget and KKT residuals are evaluated in float64 on the applied weights.

Table 5: Numerical tolerances for the PeerCredit allocation after float32 casting.

Quantity	Numerical tolerance
Budget residual	$\left\lvert \sum_i m_i w_i - \beta_p \right\rvert \le 10^{-6}$
KKT residual	$\max_i \lvert w_i - \mathrm{clip}(\beta_p - \rho c_i - \lambda_p, 0.1, 1) \rvert \le 10^{-6}$

B.4 Mixed Precision and Computational Scope

The factor-extraction prepass performs one additional teacher-forced actor forward per response. It captures the final-block MLP input a_i and the residual stream z_i immediately before the final RMSNorm. Batches of tokens then reconstruct the target log-probabilities through the final RMSNorm

Algorithm 1 PeerCredit

Require: Frozen outer batch with rewards R, base advantages A, old-policy token probabilities, prompt-group identities, and canonical problem keys; rollout count G, allocation coefficient ρ, reference radius r_0, and positive-reward scale α_+

Compute the token-mean loss coefficients α_i from the fixed PPO microbatch assignment.

Initialize $w_i^{\mathrm{PC}} \leftarrow 1$ for every valid response token.

Set $w_i^{\mathrm{PC}} \leftarrow \alpha_+$ for tokens in responses with $R_{pj} > 0$.

Under the frozen old policy, extract (e_i, a_i) for every valid response token and represent its selected-block score gradient as

$$G_i = e_i a_i^\top = \nabla_U \log \pi_{\theta_0}(y_i \mid s_i),$$

where U is the final-block MLP `down_proj`.

for each prompt group h **do**

Compute the leave-one-out reward contrasts

$$\Delta R_{hj}^{\mathrm{loo}} \leftarrow R_{hj} - \frac{1}{G-1} \sum_{j' \neq j} R_{hj'}.$$

Form the reward-weighted peer direction

$$H_h \leftarrow \frac{1}{G} \sum_{j=1}^{G} \Delta R_{hj}^{\mathrm{loo}} \sum_t G_{hjt}.$$

for each target prompt group p **do**

Construct the eligible negative-token set

$$\mathcal{N}_p \leftarrow \{i = (p, j, t) : R_{pj} \leq 0,\ A_i < 0\}.$$

if $\mathcal{N}_p \neq \varnothing$ **then**

Compute the base negative-credit coefficients and normalized masses

$$q_i \leftarrow \alpha_i |A_i|, \qquad m_i \leftarrow \frac{q_i}{\sum_{k \in \mathcal{N}_p} q_k}, \qquad i \in \mathcal{N}_p.$$

Construct the probability-derived reference budget β_p using Equations 32–36.

Define the eligible peer set

$$\mathcal{P}_p \leftarrow \{h : \mathrm{key}(x_h) \neq \mathrm{key}(x_p)\}.$$

if $\mathcal{P}_p = \varnothing$ **then**

Set $w_i^{\mathrm{PC}} \leftarrow \beta_p$ for all $i \in \mathcal{N}_p$.

else

Compute token–peer agreement scores

$$S_{ih} \leftarrow \langle G_i, H_h \rangle_F = e_i^\top H_h a_i, \qquad i \in \mathcal{N}_p,\ h \in \mathcal{P}_p.$$

Construct the five token features in Equation 26 and residualize each peer column using the weighted regression in Equation 27.

Aggregate and normalize the residual peer evidence using Equations 29–30 to obtain token-level peer support c.

Solve the budget-constrained allocation in Equation 9 for $w^\star$.

Set $w_i^{\mathrm{PC}} \leftarrow w_i^\star$ for all $i \in \mathcal{N}_p$.

Detach the token weights and form

$$\widetilde{A}_i = \mathrm{sg}(w_i^{\mathrm{PC}}) A_i.$$

Optimize the clipped GRPO policy objective in Equation 58 using $\widetilde{A}_i$.

and LM head and differentiate only with respect to a detached copy of z_i to obtain e_i. All valid response tokens are retained.

The model forward uses bfloat16 autocast. RMS statistics and log-sum-exp operations are evaluated in float32. Gradient factors are transferred to CPU; peer accumulation and bilinear contractions use float32 with TF32 disabled. Residualization and allocation are performed in float64, followed by float32 returned weights.

With B_{active} stored peer matrices, peer-matrix storage scales as

$$4B_{\text{active}}d_{\text{model}}d_{\text{ff}} \tag{64}$$

bytes per rank in float32. For the Qwen3-4B 2560×9728 parameter block and at most 256 peer groups, this corresponds to approximately 23.75 GiB per rank for peer matrices alone. Factor storage scales with the number of valid response tokens times $(d_{\text{model}} + d_{\text{ff}})$. PeerCredit does not increase the rollout budget; its additional computation comes from the teacher-forced prepass, local differentiation, peer aggregation, residualization, and allocation.

C Experimental Details and Additional Results

C.1 Mathematical Reasoning Configuration

Table 6 summarizes the training configuration for the mathematical reasoning experiments. All compared methods use the same RLVR training protocol and differ in the credit-weighting rule under comparison.

Table 6: Training configuration for the mathematical reasoning experiments.

Item	Value
Model / training data	Qwen3-1.7B, Qwen3-4B, or Qwen3-8B / DAPO-Math-17k
Prompt groups / rollouts per group	256 / 4
Training steps	200 (main experiments); 100 (sensitivity analysis)
Prompt / response token limits	2048 / 4096
Training temperature / top-p / top-k	1.0 / 1.0 / -1
Thinking template	`enable_thinking=false`
Advantage estimator	GRPO with within-group reward normalization
Learning rate / warmup steps	10^{-6} / 10
Weight decay / gradient-norm clip	0.1 / 1.0
PPO epochs / actor shuffle	1 / disabled
PPO minibatch / per-GPU microbatch sizes	64 / 4
Loss aggregation / dynamic batching	Token mean / disabled
PPO lower / upper clip widths	0.2 / 0.2
Entropy / actor-KL coefficients	0 / 0
Rollout dtype / actor strategy	bfloat16 / FSDP
Shared gradient block	Final MLP down-projection
Score features / allocation equalities	Five / one (budget conservation)
Positive-reward response scale	0.1
Eligible negative-weight bounds	$[0.1, 1]$
Reference radius r_0 / allocation coefficient ρ	0.2 / 0.2

For evaluation, each problem is sampled independently 16 times with temperature 0.6, top-$p = 0.95$, top-$k = 50$, and a maximum generation length of 8,192 tokens, with `enable_thinking=false`. The decoder seed is fixed to 0 independently of the training seed. For benchmark b with N_b problems and binary verifier output $v_{bns} \in \{0, 1\}$, we report

$$\text{Avg@16}_b = \frac{100}{16N_b} \sum_{n=1}^{N_b} \sum_{s=1}^{16} v_{bns}. \tag{65}$$

C.2 TOOL-USE CONFIGURATION

Tool-use experiments initialize from Qwen3-8B and use the `rlla_4k` tool-instruction dataset. We use 3,920 examples for training and reserve 80 examples as a held-out validation split. Rollouts are generated with thinking enabled. Table 7 summarizes the corresponding training configuration.

Table 7: Training configuration for the tool-use experiments.

Item	Value
Model / training data	Qwen3-8B / `rlla_4k`
Training / validation examples	3,920 / 80
Prompt groups / rollouts per group	512 / 4
Training steps	105
Prompt / response token limits	2048 / 2048
Training temperature / top-p / top-k	1.0 / 1.0 / -1
Thinking template	`enable_thinking=true`
Advantage estimator	GRPO with within-group reward normalization
Learning rate / warmup steps	10^{-6} / 0
Weight decay / gradient-norm clip	0.01 / 1.0
PPO epochs	1
PPO minibatch / per-GPU microbatch sizes	128 / 4
PPO lower / upper clip widths / dual clip	0.2 / 0.2 / 3
Entropy / actor-KL coefficients	0 / 0
Positive-reward response scale	0.1
Eligible negative-weight bounds	$[0.1, 1]$

BFCL and API-Bank are evaluated at the final training checkpoint using their task-specific official evaluators.

D ABLATION AND SENSITIVITY EXPERIMENTS

D.1 ABLATION DEFINITIONS

All ablations use the same training pipeline and retain the reward and advantage definitions, 0.1 positive-reward response scale, probability-derived β_p, negative-token eligibility, $[0.1, 1]$ weight bounds, $\rho = 0.2$, rollout budget, and batching protocol unless the intervention explicitly targets one of these components.

PeerCredit. The full method uses exact final down-projection gradient factors, leave-one-out peer directions, five-feature score residualization, and the single budget-preserving quadratic allocation described in Section 4.

w/o Cross-Prompt Credit Exchange. Use $c = 0$, which gives $\delta = 0$ and $w_i = \beta_p$ for all eligible negative tokens. This yields the matched-control variant. It retains the same reference attenuation and positive-reward response scale while eliminating peer-dependent allocation. The rollout budget and reference construction are identical to the full method.

w/o Score Residualization. Replace the five-column score feature matrix by the intercept only. For every peer,

$$Z_{ih}^{\mathrm{center}} = S_{ih} - \sum_r m_r S_{rh}, \qquad c_i^{\mathrm{center}} = \frac{\sum_h \omega_h Z_{ih}^{\mathrm{center}}}{\sqrt{\sum_{rh} m_r \omega_h (Z_{rh}^{\mathrm{center}})^2}}, \tag{66}$$

with the same zero-energy convention and relative roundoff rule. This removes the two probability features, relative position, and log length, but preserves weighted centering and pooled-energy normalization. The budget-preserving equality constraint remains unchanged.

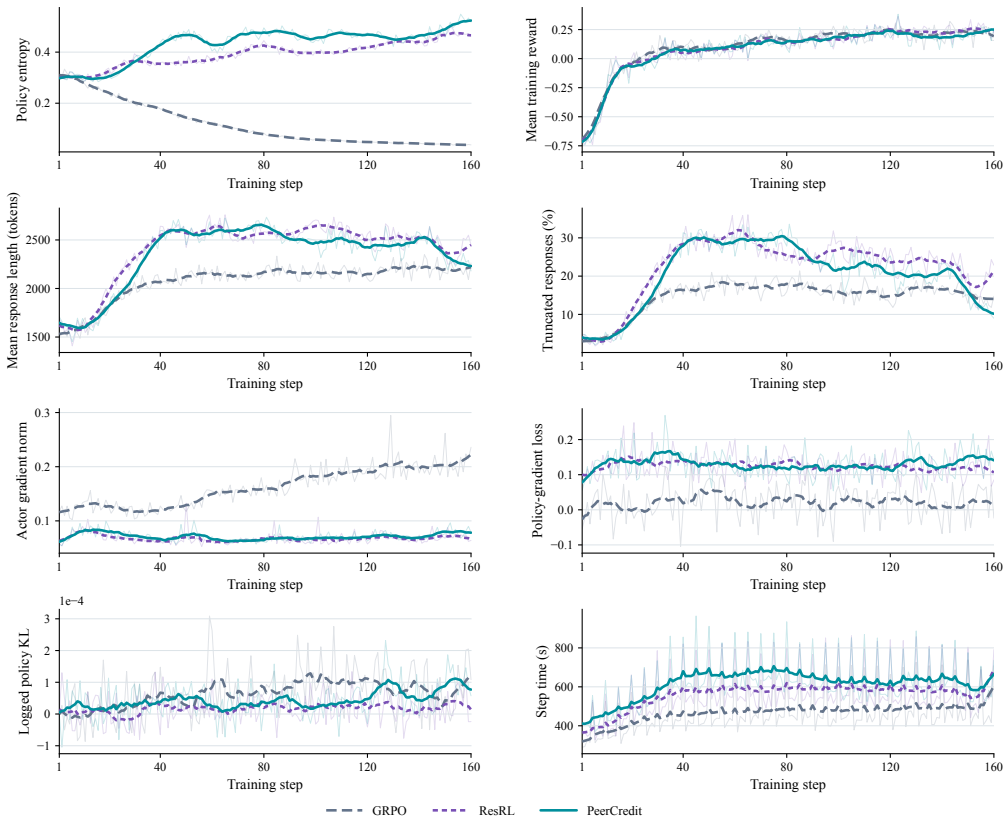


Figure 6: Training dynamics of GRPO, ResRL, and PeerCredit on Qwen3-4B over training steps. The panels show policy entropy, mean training reward, mean response length, truncation rate, actor gradient norm, policy-gradient loss, logged policy KL, and step time. Faint lines denote per-step measurements; prominent lines are 10-step moving averages.

Shuffled Peer Rewards. For each peer group h, choose a permutation π_h of its G response indices and use

$$R_{hj}^{\text{review}} = R_{h,\pi_h(j)}, \qquad H_h^{\text{shuf}} = \frac{1}{G}\sum_j \left(R_{hj}^{\text{review}} - \frac{1}{G-1}\sum_{j' \neq j} R_{hj'}^{\text{review}} \right) \sum_t G_{hjt}. \tag{67}$$

We permute reward assignments within each peer group, preserving the reward multiset while disrupting the trajectory–reward correspondence. A single permutation is held fixed within each outer batch. Target-side advantages, success/failure indicators, masks, and reference budgets continue to use the original rewards R_{hj}. All-equal peer rewards remain zero.

w/o Within-Response Allocation. First compute the full method's eligible weights. For response (p, j) with a nonempty eligible subset $\mathcal{N}_{pj}$, replace only those weights by

$$\bar{w}_{pj} = \frac{\sum_{i \in \mathcal{N}_{pj}} q_i w_i}{\sum_{i \in \mathcal{N}_{pj}} q_i}, \qquad w_i^{\text{resp}} = \bar{w}_{pj} \quad (i \in \mathcal{N}_{pj}). \tag{68}$$

Empty subsets are unchanged. We use the q_i-weighted mean within each response. It removes within-response variation but preserves each response's allocated negative-credit amount and therefore the full prompt budget. A convex average remains in $[0.1, 1]$. Because the reference is constant within the prompt, conditional Jensen gives $\|w^{\text{resp}} - b\|_m \leq \|w - b\|_m$.

D.2 Allocation Strength and Peer-Count Sensitivity

All entries in Tables 8 and 9 use Qwen3-4B, training seed1001, and the fixed step100 checkpoint. The rollout batch remains 256 prompts with four responses per prompt. We evaluate 16 responses per problem with temperature 0.6, top-$p = 0.95$, top-$k = 50$, a response limit of 8,192 tokens, and no-thinking serialization. The six-benchmark average gives equal weight to each benchmark.

Allocation strength. We vary ρ in the quadratic allocation while holding $r_0 = 0.2$ and using all eligible peers. The reference negative-credit budget is constructed by the same rule in every run. Thus ρ controls the trade-off between peer support and quadratic deviation from the reference allocation.

Table 8: Allocation-strength sensitivity at step100 (Avg@16, %). All rows use seed1001 and the same 100-step training budget.

Setting	AIME 2024	AIME 2025	AMC 2023	MATH-500	Minerva Math	Olympiad Bench	Avg.
$\rho = 0.10$	42.08	36.04	86.72	81.58	38.10	54.01	56.42
$\rho = 0.30$	39.58	30.00	78.59	87.66	39.34	60.88	56.01
$\rho = 0.40$	43.33	37.71	81.41	89.53	42.69	63.07	59.62

Peer count. Here M denotes the maximum number of eligible peer prompt groups included for each target. We hold $\rho = r_0 = 0.2$. For each target, same-key prompts are excluded. Peer subsets are selected deterministically using a reward-independent hash of selection seed 96051, the target key, and the peer key. The first $\min(M, |\mathcal{P}_p|)$ groups define the peer panel, yielding nested peer subsets for a fixed outer batch. Zero-gradient peers remain in the panel and its normalization. The default method uses all eligible prompt groups, at most 255 in a 256-prompt batch before duplicate-key exclusion.

Table 9: Peer-count sensitivity at step100 (Avg@16, %). These are single-training-seed results; $\rho = r_0 = 0.2$ throughout.

Setting	AIME 2024	AIME 2025	AMC 2023	MATH-500	Minerva Math	Olympiad Bench	Avg.
$M = 8$	42.50	40.00	83.44	80.11	32.84	57.92	56.13
$M = 16$	34.38	29.58	79.69	89.26	40.10	60.07	55.51
$M = 32$	31.04	33.33	78.75	83.94	38.17	54.30	53.26
$M = 64$	32.08	28.12	70.62	84.26	37.02	56.44	51.42
$M = 128$	38.75	33.96	72.50	87.05	41.54	57.64	55.24

The measured dependence on M is non-monotonic. In particular, the $M = 8$ run has the highest six-benchmark average among the displayed caps, while $M = 16$ has the highest MATH-500 score. These experiments characterize the end-to-end sensitivity of PeerCredit to peer count.

D.3 Per-Seed Mathematical Results

Table 10 reports step-200 results for training seeds 1001–1003 on Qwen3-1.7B, Qwen3-4B and Qwen3-8B. Reported standard deviations are sample standard deviations across training seeds, with evaluation decoder seed fixed at 0.

Gaokao 2023 English is evaluated separately from the common six-benchmark suite. At seed1001, step200, Qwen3-4B obtains 78.17% and Qwen3-8B obtains 79.59%; their corresponding seven-benchmark averages are 64.59% and 63.37%.

Table 10: Per-training-seed mathematical results at step200 (Avg@16, %). Avg. is the six-benchmark mean. The final row of each model reports the mean and sample standard deviation across training seeds.

Setting	AIME 2024	AIME 2025	AMC 2023	MATH-500	Minerva Math	Olympiad Bench	Avg.
1.7B, 1001	28.96	25.83	73.75	86.54	36.83	58.68	51.77
1.7B, 1002	25.42	25.63	69.53	85.30	37.16	57.37	50.07
1.7B, mean ± std.	27.19 ± 2.50	25.73 ± 0.15	71.64 ± 2.98	85.92 ± 0.88	36.99 ± 0.23	58.03 ± 0.92	$\mathbf{50.92 \pm 1.20}$
4B, 1001	45.42	40.00	90.47	91.06	42.83	64.16	62.32
4B, 1002	43.13	38.54	87.34	83.51	39.84	59.40	58.63
4B, 1003	42.50	37.71	86.41	87.80	37.87	57.24	58.25
4B, mean ± std.	43.68 ± 1.54	38.75 ± 1.16	88.07 ± 2.13	87.46 ± 3.79	40.18 ± 2.50	60.27 ± 3.54	$\mathbf{59.73 \pm 2.25}$
8B , 1001	56.04	45.42	88.91	93.14	45.59	70.35	66.57
8B , 1002	53.34	42.08	87.66	92.15	44.35	67.92	64.58
8B , 1003	50.42	37.92	90.47	93.88	45.13	69.79	64.60
8B, mean ± std.	53.27 ± 2.81	41.81 ± 3.76	89.01 ± 1.41	93.06 ± 0.87	45.02 ± 0.63	69.35 ± 1.27	$\mathbf{65.25 \pm 1.14}$

Table 11: Per-training-seed API-Bank results for PeerCredit-8B (%). All runs use positive-reward scale $\alpha_+ = 0.10$. The final row reports the mean and sample standard deviation across the three seeds.

Configuration / seed	Level 1 Accuracy	Level 2 Accuracy	Level 3 Accuracy	Overall Accuracy
$\alpha_+ = 0.10$, seed1003	74.19	64.18	44.27	66.50
$\alpha_+ = 0.10$, seed1004	75.44	65.67	49.62	68.68
$\alpha_+ = 0.10$, seed1001	75.19	64.18	47.33	67.84
Mean ± std.	74.94 ± 0.66	64.68 ± 0.86	47.07 ± 2.68	$\mathbf{67.67 \pm 1.10}$

Table 12: Per-run BFCL results for PeerCredit-8B (%). MT: multi-turn; MF: Missing Functions; MP: Missing Parameters; LC: Long Context; ST: single-turn. The final row reports the mean and sample standard deviation across the three runs.

Seed	MT Overall	Base	MF	MP	LC	ST Non-Live	ST Live	Overall
1001	41.00	51.00	44.50	35.50	33.00	89.46	78.76	69.08
1003	40.38	52.00	43.00	34.00	32.50	88.58	79.83	69.19
1004	38.88	47.00	42.00	32.50	34.00	89.12	80.32	68.91
Mean ± std.	40.09±1.09	50.00±2.65	43.17±1.26	34.00±1.50	33.17±0.76	89.05±0.44	79.64±0.80	**69.06±0.14**

E EVALUATION PROMPTS

The following templates cover the six mathematical reasoning benchmarks, BFCL, and API-Bank. Double-bracket placeholders denote dataset-dependent fields and are not literal instructions. Message-role labels appear in box titles, not in the message content. Mathematical evaluation disables thinking through the chat-template argument `enable_thinking=false`; the step-by-step instruction is retained exactly as stored in the data.

AIME 2024: user message

```
Solve the following math problem step by step. The last line of your
    response should be of the form Answer: $Answer (without quotes)
    where $Answer is the answer to the problem.

[[MATH_PROBLEM]]

Remember to put your answer on its own line after "Answer:".
```

AIME 2025: user message

```
Solve the following math problem step by step. The last line of your
    response should be of the form Answer: $Answer (without quotes)
    where $Answer is the answer to the problem.

[[MATH_PROBLEM]]

Remember to put your answer on its own line after "Answer:".
```

AMC 2023: user message

```
Solve the following math problem step by step. The last line of your
    response should be of the form Answer: $Answer (without quotes)
    where $Answer is the answer to the problem.

[[MATH_PROBLEM]]

Remember to put your answer on its own line after "Answer:".
```

MATH-500: user message

```
Solve the following math problem step by step. The last line of your
    response should be of the form Answer: $Answer (without quotes)
    where $Answer is the answer to the problem.

[[MATH_PROBLEM]]

Remember to put your answer on its own line after "Answer:".
```

Minerva Math: user message

```
Solve the following math problem step by step. The last line of your
    response should be of the form Answer: $Answer (without quotes)
    where $Answer is the answer to the problem.
```

Minerva Math: user message (continued)

```
[[MATH_PROBLEM]]

Remember to put your answer on its own line after "Answer:".
```

OlympiadBench: user message

```
Solve the following math problem step by step. The last line of your
    response should be of the form Answer: $Answer (without quotes)
    where $Answer is the answer to the problem.

[[MATH_PROBLEM]]

Remember to put your answer on its own line after "Answer:".
```

BFCL. The selected Qwen function-calling handler combines the benchmark system instruction with its tool-format instruction, separated by two newline characters. The two components are shown separately below to make this composition explicit; neither is the tool-training prompt. Benchmark-provided system context and previous conversation turns are retained when present. Tool feedback is rendered inside `<tool_response>` tags.

BFCL: benchmark system instruction

```
You are an expert in composing functions. You are given a question
    and a set of possible functions. Based on the question, you will
    need to make one or more function/tool calls to achieve the
    purpose.
If none of the functions can be used, point it out. If the given
    question lacks the parameters required by the function, also
    point it out.
You should only return the function calls in your response.

If you decide to invoke any of the function(s), you MUST put it in
    the format of [func_name1(params_name1=params_value1,
    params_name2=params_value2...), func_name2(params)]
You SHOULD NOT include any other text in the response.

At each turn, you should try your best to complete the tasks
    requested by the user within the current turn. Continue to output
     functions to call until you have fulfilled the user's request to
     the best of your ability. Once you have no more functions to
    call, the system will consider the current turn complete and
    proceed to the next turn or task.

Here is a list of functions in JSON format that you can invoke.
[[FUNCTION_DESCRIPTIONS]]
```

BFCL: appended Qwen tool-format instruction

```
# Tools

You may call one or more functions to assist with the user query.
```

BFCL: appended Qwen tool-format instruction (continued)

```
You are provided with function signatures within <tools></tools> XML
    tags:
<tools>
[[FUNCTION_SIGNATURES_AS_JSON_LINES]]
</tools>

For each function call, return a json object with function name and
    arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

BFCL: user message

```
[[BFCL_USER_QUERY]]
```

API-Bank. The evaluator reads the processed benchmark's `system` and `user` fields and forwards them to `llm.chat` in that order. The fixed system text below is shared by all 597 processed records across levels 1–3; only the available tool definitions vary. The user-message placeholder preserves the dataset's dialogue-history heading and tagged conversation verbatim. All records append the one-tool-call instruction shown below. These instructions are already present in the processed data; the evaluator does not add them at generation time.

API-Bank: system message

```
You are a helpful multi-turn dialogue assistant capable of leveraging
     tool calls to solve user tasks and provide structured chat
    responses.

**Available Tools**
In your response, you can use the following tools:
[[AVAILABLE_TOOLS]]

**Steps for Each Turn**
1. **Think:** Recall relevant context and analyze the current user
    goal.
2. **Decide on Tool Usage:** If a tool is needed, specify the tool
    and its parameters.
3. **Respond Appropriately:** If a response is needed, generate one
    while maintaining consistency across user queries.

**Output Format**
'''plaintext
<think> Your thoughts and reasoning </think>
<tool_call>
{"name": "Tool name", "parameters": {"Parameter name": "Parameter
    content", "... ...": "... ..."}}
{"name": "... ...", "parameters": {"... ...": "... ...", "... ...":
    "... ..."}}
...
</tool_call>
<response> AI's final response </response>
'''

**Important Notes**
```

API-Bank: system message (continued)

```
1. You must always include the `<think>` field to outline your
    reasoning. Provide at least one of `<tool_call>` or `<response>`.
     Decide whether to use `<tool_call>` (possibly multiple times), `<
    response>`, or both.
2. You can invoke multiple tool calls simultaneously in the `<
    tool_call>` fields. Each tool call should be a JSON object with a
     "name" field and an "parameters" field containing a dictionary
    of parameters. If no parameters are needed, leave the "parameters
    " field an empty dictionary.
3. Refer to the previous dialogue records in the history, including
    the user's queries, previous `<tool_call>`, `<response>`, and any
     tool feedback noted as `<obs>` (if exists).
```

API-Bank: user message

```
[[CONVERSATION_HISTORY_WITH_DATASET_HEADING]]

<user> Based on our conversation above, please only make one tool
    call to solve my need.</user>
```