Beyond the Sampled Token: Preserving Candidate Support in RLVR

Ruotian Peng[1,2,†] **Yi Ren**[3,†] **Zhouliang Yu**[4] **Weiyang Liu**[4] **Yandong Wen**[2,*]
[1]Zhejiang University [2]Westlake University
[3]University of Oxford [4]The Chinese University of Hong Kong
[†]Equal contribution [*]Corresponding author Spherelab.ai/CaSP
wenyandong@westlake.edu.cn

Abstract

We revisit exploration collapse in reinforcement learning with verifiable rewards (RLVR), from the perspective of the *candidate distribution* for next-token prediction. We formally show that as probability concentrates on the top-1 candidate, the expected number of distinct responses collapses to one regardless of the sampling budget K. This theoretical implication is further verified by our empirical tracking of top-N candidate probabilities during training, where the top-1 candidate progressively dominates while plausible alternatives are suppressed. These findings suggest a key desideratum for effective exploration: *preserving non-negligible probability mass on the top-N candidates*. To this end, we propose Candidate-aware Support Preservation (CaSP), with two complementary designs. Specifically, CaSP redistributes positive gradients among top-N candidates for correct responses, and applies a stronger penalty to the top-1 candidate for incorrect responses. Unlike many exploration-oriented methods that improve pass@K at the cost of pass@1, CaSP improves pass@K across the full K spectrum. These gains generalize to 6 math, 2 logical-reasoning, and 2 coding benchmarks, and scales to 32B-parameter models and sampling budgets up to $K = 1024$, positioning it as a principled, candidate-level approach for RLVR exploration.

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) offers a simple recipe for improving LLM reasoning: the model generates responses, and updates itself by rewarding correct ones and penalizing incorrect ones (Shao et al., 2024; Schulman et al., 2017; Hu, 2025). This coupled process induces a bias towards exploitation over exploration, whereby the model collapses to a narrow set of responses (Liang et al., 2025; He et al., 2025a). Such an effect is evident in improved pass@1, which measures how often a single response is correct, but degraded pass@K (K>1), which measures whether the model can find correct solutions with many attempts (Yue et al., 2025; Wu et al., 2025). Without sufficient exploration, the model struggles with more challenging scenarios, ultimately capping its potential for further reasoning improvement (Chen et al., 2025b; Song et al., 2025).

To address this exploration deficit, existing approaches modify the optimization signal at different granularities, with conceptual comparison in Figure 1. *Response-level* advantage shaping reweights entire responses, either through pass@K objectives (Chen et al., 2025b; Walder and Karkhanis, 2026) or through response rarity, diversity, and correctness signals (Liu et al., 2026; Gai et al., 2025; Zhu et al., 2025). These methods improve selection among sampled responses but do not explicitly encourage exploring new ones. *Token-level* advantage shaping operates at each decoding step, typically through entropy-based bonuses computed from the next-token distribution (Cheng et al., 2025; Cui et al., 2025; Jiang et al., 2025). However, entropy is too coarse to distinguish which candidates deserve probability mass, and may push it onto unrelated ones. Sitting between these levels, *segment-level* approaches shape advantages over contiguous token spans (Chen et al., 2025a). *Hybrid* approaches combine response- and token-level signals, through curiosity-driven or outcome-conditioned rewards (Dai et al., 2025; Song et al., 2025). Overall, existing approaches either insufficiently expand the response space or expand it without control, falling short of effective exploration.

We take an alternative perspective grounded in next-token prediction. At each decoding step, an LLM outputs a candidate distribution over its vocabulary[1], offering a fine-grained view of its ex-

[1]The *vocabulary* is the set of all possible *candidates*; one *candidate* is sampled and becomes the *token* in a *response*.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 13318–13340
October 24-29, 2026 ©2026 Association for Computational Linguistics

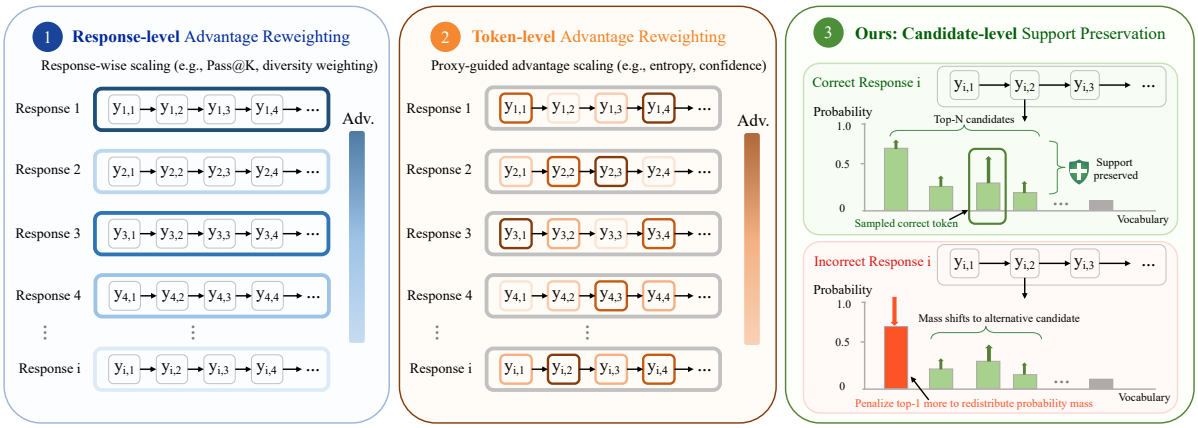


Figure 1: Overview of exploration-oriented methods at different intervention granularities. Existing approaches mainly operate at the response level or token level, whereas CaSP operates at the candidate level by explicitly preserving probability mass among top-N plausible next-token candidates.

ploration behavior. How probability is distributed across candidates determines whether the model explores multiple reasoning paths or collapses into a single deterministic response (Deng et al., 2025b; Zhu et al., 2025), as formally stated in Proposition 3. Yet capturing the full distributioni is computationally prohibitive since modern vocabularies exceed 100K candidates. This likely explains why prior work favored scalar measures like entropy.

We sidestep this constraint by examining candidate distributions empirically. These distributions are highly skewed (Figure 2), with only a few candidates carrying non-negligible probability mass, so the top-N candidates serve as a tractable yet faithful approximation of the full distribution. During RLVR fine-tuning, we observe that probability mass progressively concentrates on the top-1 candidate while plausible alternatives are suppressed. This pattern matches the Proposition 3 and directly explains the pass@K degradation. These findings point to a clear desideratum for effective exploration: *preserving non-negligible probability mass on the top-N candidates*.

To achieve this desideratum, we propose Candidate-aware Support Preservation (CaSP), with two complementary designs. For correct responses, CaSP redistributes positive gradients among the top-N candidates, smoothing reinforcement across plausible alternatives rather than concentrating it on the sampled token. For incorrect responses, CaSP applies a stronger penalty to the top-1 candidate to counteract the squeeze effect, redirecting probability mass to alternatives. These updates prove especially effective when applied to high-entropy tokens in the reasoning path.

We evaluate CaSP on multiple LLM backbones and a diverse set of benchmarks, including 6 math, 2 logical reasoning, and 2 coding tasks. Unlike GRPO (which improves pass@1 but struggles at pass@K) and exploration-oriented baselines (which improve pass@K but sacrifice pass@1), CaSP achieves the best of both worlds, improving pass@K across the full spectrum (Figure 2). Moreover, these gains scale to models up to 32B parameters and sampling budgets up to $K = 1024$. In summary, our work makes three contributions:

- We introduce a candidate-distribution perspective on RLVR exploration, supported by both formal analysis and empirical verification.
- We propose CaSP, a principled candidate-level method that encourages exploration by preserving top-N candidate supports.
- CaSP's gains generalize to multiple LLM backbones and diverse reasoning benchmarks, scaling to 32B models and $K = 1024$ sampling budgets.

2 Background and Preliminaries

RLVR trains an LLM using rewards from a programmatic verifier (*e.g.,* a math grader or code checker). Group relative policy optimization (GRPO) (Shao et al., 2024), is a widely used RLVR method tailored for LLM post-training. Given a question x, the model generates G different responses $\{y_i\}_{i=1}^{G}$ and updates its parameters θ as:

$$\mathcal{J}_{\text{GRPO}}(\theta;\gamma_{i,t}) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\Big[\min\big(\gamma_{i,t}A_i, \mathsf{clip}_\epsilon(\gamma_{i,t})A_i\big) - \beta\,\mathbb{D}_{\text{KL}}(\pi_\theta\|\pi_{\text{ref}})\Big]$$

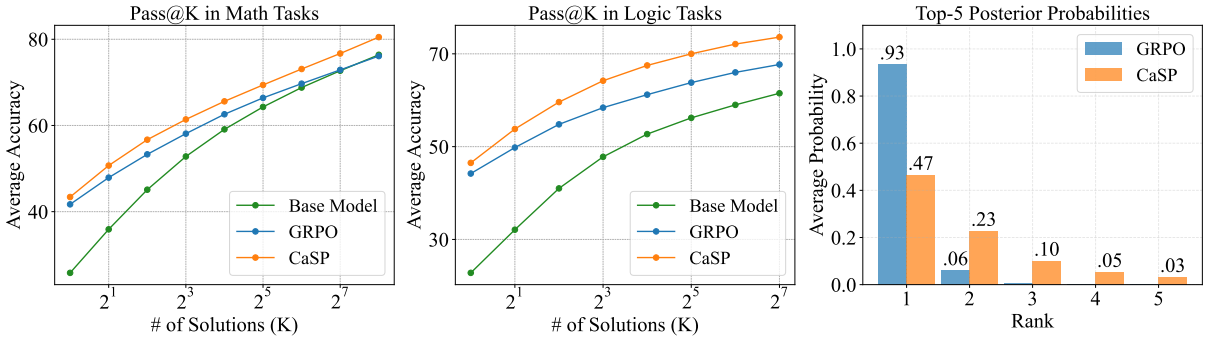


Figure 2: **Left/middle**: pass@K of CaSP vs. GRPO on math (AIME24/25, AMC, MATH500, Minerva, OlympiadBench) and logic (SynLogic, BBH) benchmarks; CaSP improves across the K spectrum. **Right**: average probability of the rank-n candidate; CaSP's distribution is less concentrated than GRPO's.

where $\text{clip}_\epsilon(\gamma)$ truncates the ratio γ to $[1-\epsilon, 1+\epsilon]$, and $\gamma_{i,t} = \pi_\theta(y_{i,t}|s_{i,t})/\pi_{\text{old}}(y_{i,t}|s_{i,t})$ is the likelihood ratio between the current policy π_θ and the previous-iteration policy π_{old} at the t-th token of response y_i, with $s_{i,t} = (x, y_{i,<t})$ denoting the decoding state. The KL term uses π_{ref}, the frozen pre-trained reference. The advantage $A_i = (r_i - \text{mean}(\{r_j\}_{j=1}^G))/\text{std}(\{r_j\}_{j=1}^G)$ is computed by normalizing the verifier reward $r_i \in \{0, 1\}$ of response y_i across the group of G rollouts.

GRPO can be analyzed in gradient space under the learning dynamics framework of (Ren and Sutherland, 2025). The dominant contribution to parameter updates comes from $\nabla_\theta A_i \gamma_{i,t}$. Here, the advantage A_i can be viewed as a response-level adaptive learning rate that scales gradients from different responses y_i. Ignoring the KL and clipping terms (both primarily introduced for stability), the main optimization direction is:

$$\nabla_\theta A_i \gamma_{i,t} = A_i \cdot \text{sg}(\gamma_{i,t}) \cdot \nabla_\theta \log \pi_\theta(y_{i,t}|s_{i,t}), \quad (1)$$

where $\text{sg}(\cdot)$ is the stop-gradient operator. CaSP builds on this gradient form to redistribute mass across the candidate distribution.

3 A Candidate-Level Analysis of Why RLVR Concentrates Probability

In this section, we show that (i) both positive and negative RLVR updates push probability mass toward the rank-1 candidate, (ii) this candidate-level concentration leads to low-diversity responses, and (iii) empirical tracking rank-wise log-probabilities across RLVR methods reproduces the predicted concentration. These results highlight the importance of preserving supports over the top-N candidates, which we take as a natural desideratum for effective exploration.

3.1 RLVR Updates Reinforce Concentration

We now show why on-policy RLVR updates natually push candidate distribution toward concentration, considering positive and negative samples separately. Let $\mathcal{V}$ denote the vocabulary; for any candidate $v \in \mathcal{V}$ at state $s_{i,t}$, let z_v and $p_v = \pi_\theta(v|s_{i,t})$ be its logit and probability.

Observation 1 (Positive updates amplify dominant candidates)**.** *At state $s_{i,t}$, GRPO samples tokens from $\pi_\theta(\cdot|s_{i,t})$, so the expected positive-advantage reinforcement of each candidate is proportional to its current probability p_v.*

Since the rank-1 candidate is sampled most frequently and therefore reinforced most by positive advantages. The training process widens the probability gap between the rank-1 candidate and lower-ranked alternatives, driving the next-token distribution toward over-concentration.

Prior work (Zhu et al., 2025) suggests that negative samples could counteract concentration by suppressing incorrect token and redistributing probability mass to alternatives (Zhu et al., 2025). However, this redistribution flows disproportionately to high-probability candidates, known as the squeeze effect (Ren and Sutherland, 2025).

Lemma 2 (Squeeze effect of negative updates)**.** *For the unclipped GRPO term $A_i\gamma_{i,t}$ with $A_i < 0$, the gradient decreases the sampled token's probability $p_{y_{i,t}}$. For any alternative $v \neq y_{i,t}$, the softmax coupling gives:*

$$\frac{\partial p_{y_{i,t}}}{\partial z_v} = -p_{y_{i,t}} p_v.$$

The negative update therefore raises z_v in proportion to $p_{y_{i,t}} p_v$. If the penalized token $y_{i,t}$ is not the rank-1 candidate, the rank-1 candidate absorbs the largest increase. (See Appendix B.1.)

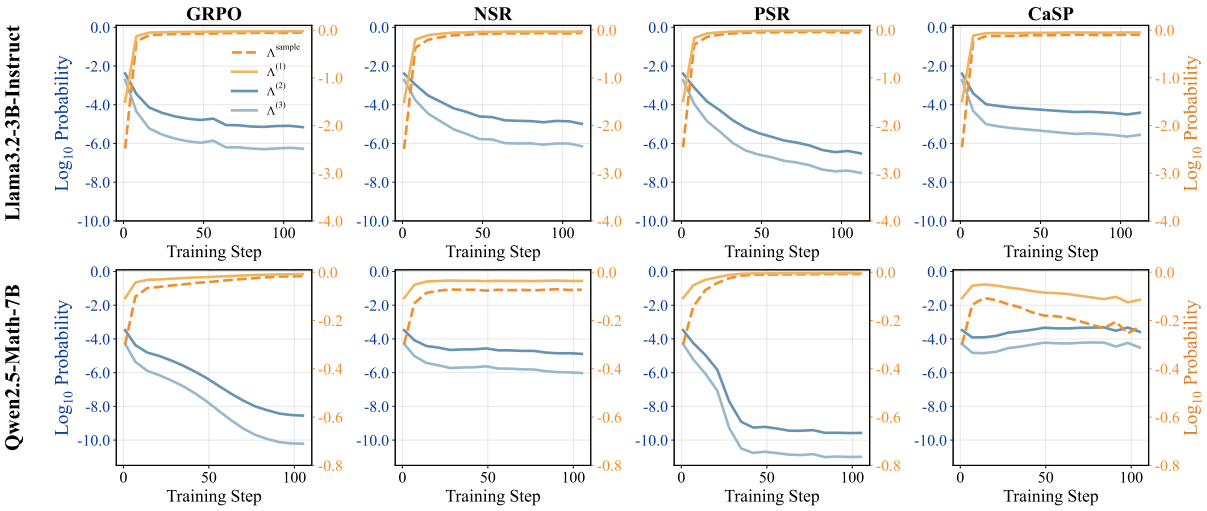


Figure 3: Training dynamics of the sampled-token log-probability Λ^{sample} and rank-n candidate log-probabilities $\Lambda^{(n)}$ across RLVR variants: GRPO, PSR (training with only positive samples), NSR (training with only negative samples), and CaSP. Following the setup of (Zhu et al., 2025), we train Llama3.2-3B-Instruct on a mixture of GSM8K and MATH (Level 1) and Qwen2.5-Math-7B on the MATH dataset.

3.2 From Candidate to Response Collapse

Positive and negative updates push mass to the rank-1 candidate. We now connect this concentration to pass@K through a simple sampling fact.

Proposition 3 (Local concentration limits rollout diversity)**.** *At state $s_{i,t}$ with distribution p over $\mathcal{V}$, consider K independent rollouts that reach this state and sample their next tokens $\{\tilde{y}_{1,t}, \ldots, \tilde{y}_{K,t}\}$. The expected number of distinct next tokens is*

$$\mathbb{E}\left[|\{\tilde{y}_{1,t}, \ldots, \tilde{y}_{K,t}\}|\right] = \sum_{v \in \mathcal{V}} \left(1 - (1 - p_v)^K\right).$$

If the rank-1 candidate probability $p_{v^{(1)}}$ converges to 1, this expectation converges to 1 for any fixed sampling budget K.

Proposition 3 connects candidate-level concentration to rollout-level exploration. When the distribution at a branching state collapses to one dominant candidate, many independent rollouts choose the same next token and follow similar reasoning paths. Local loss of token diversity therefore leads to homogeneous trajectories, reducing the marginal benefit of increasing the sampling budget K.

3.3 Empirical Verification

We empirically verify the concentration by tracking rank-wise log-probabilities during training. For rank-n, we measure the average log-probability of the rank-n candidate $v_{i,t}^{(n)}$ at state $s_{i,t}$:

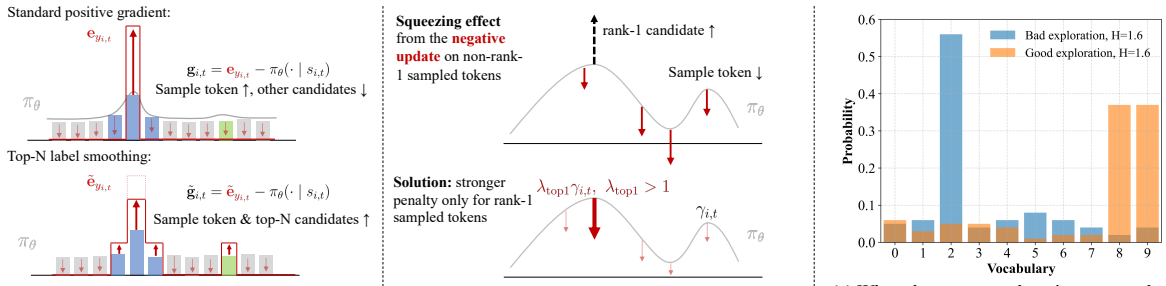
$$\Lambda^{(n)} := \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \log \pi_\theta(v_{i,t}^{(n)} \mid s_{i,t}). \quad (2)$$

We similarly define Λ^{sample} by replacing $v_{i,t}^{(n)}$ with the sampled token $y_{i,t}$. Figure 3 shows how these metrics evolve under different methods.

Under GRPO, Λ^{sample} gradually moves toward $\Lambda^{(1)}$, indicating that sampled tokens increasingly coincide with the rank-1 candidate. PSR further amplifies this movement, matching the rank-1 amplification predicted by Observation 1. In contrast, NSR slows the movement but does not eliminate it, consistent with the squeeze effect described in Lemma 2. Together, these patterns show that RLVR updates progressively enlarge the gap between $\Lambda^{(1)}$ and lower-ranked $\Lambda^{(n)}$, thereby concentrating samples around the rank-1 candidate and reducing rollout-level diversity. These findings motivate the desideratum: *preserving non-negligible probability mass on the top-N candidates.*

4 Candidate-aware Support Preservation

Section 3 shows that both positive and negative RLVR updates drive candidate probabilities toward over-concentration, motivating the desideratum of preserving non-negligible probability mass on the top-N candidates. To achieve this desideratum, we propose CaSP, comprising two complementary designs: (i) top-N label smoothing for positive gradients, and (ii) stronger rank-1 penalties for negative gradients. Both components are applied only to high-entropy tokens, which often serve as "forking" points (Deng et al., 2025b; Wang et al., 2025) (selected when their entropy exceeds the q-quantile threshold τ_q within the response).



(a) Positive updates: top-N support preservation (b) Negative updates: rank-1 targeted penalty (c) Why token-entropy alone is not enough

Figure 4: Intuition of CaSP. (a) For positive samples, we redistribute the probability mass from the top-1 candidate to the top-N candidates, mitigating over-concentration. (b) For negative samples, we apply a stronger penalty only when the sampled token is the current rank-1 candidate, avoiding the squeeze effect. (c) An example of two distributions with identical entropy but distinct probability distributions.

4.1 Positive Updates on Top-N Candidates

Observation 1 indicates that positive updates tend to amplify already dominant candidates. We make this effect explicit at the gradient level. For the sampled token $y_{i,t}$, let $\mathbf{e}_{y_{i,t}}$ be its one-hot target. The standard log-likelihood gradient is

$$\mathbf{g}_{i,t} = \frac{\partial \log \pi_\theta(y_{i,t}|s_{i,t})}{\partial \mathbf{z}_{i,t}} = \mathbf{e}_{y_{i,t}} - \pi_\theta(\cdot|s_{i,t}). \quad (3)$$

As shown in Figure 4-(a), positive updates increase the sampled token's probability while suppressing alternatives, which sharpens the distribution when the sampled token is often rank-1 candidate.

To counter this effect, CaSP reallocates part of the positive update from the sampled token to other plausible candidates via a localized variant of label smoothing (Müller et al., 2019). Unlike vanilla label smoothing, which spreads mass over the full vocabulary and can allocate probability to ungrammatical or irrelevant candidates, our variant smooths only over the model's top-N candidates:

$$\tilde{\mathbf{e}}_{y_{i,t}} = (1-\alpha)\mathbf{e}_{y_{i,t}} + \frac{\alpha}{N}\sum_{n=1}^{N} \mathbf{e}_{v_{i,t}^{(n)}}, \quad (4)$$
$$\tilde{\mathbf{g}}_{i,t} = \tilde{\mathbf{e}}_{y_{i,t}} - \pi_\theta(\cdot|s_{i,t}),$$

where $\alpha \in [0,1]$ controls the smoothing strength. This definition is unchanged when the sampled token itself belongs to the top-N set, ensuring consistent treatment of the sampled token. Equivalently, this gradient can be implemented by replacing the original ratio $\gamma_{i,t}$ with

$$\gamma_{i,t}^{+} = (1-\alpha)\gamma_{i,t} + \frac{\alpha}{N}\sum_{n=1}^{N} \mathrm{sg}\left(\frac{\gamma_{i,t}}{\gamma_{i,t}^{(n)}}\right)\gamma_{i,t}^{(n)},$$

where $\gamma_{i,t}^{(n)} = \pi_\theta(v_{i,t}^{(n)}|s_{i,t})/\pi_{\theta_{\text{old}}}(v_{i,t}^{(n)}|s_{i,t})$ is the ratio evaluated on the rank-n candidate. The stop-gradient term preserves the importance-sampling coefficient of the sampled token; see Appendix A for details. Optimizing with $\gamma_{i,t}^{+}$ redistributes positive gradients across plausible top-N candidates, forming a plateau rather than a sharp peak, as illustrated in Figure 4-(a).

4.2 Negative Updates on Top-1 Candidate

Lemma 2 shows that negative updates depend on which token is penalized: penalizing rank-1 can flatten the distribution, while penalizing lower-ranked tokens can squeeze mass toward rank-1. Thus, uniformly amplifying negative terms (Zhu et al., 2025) also strengthens non-rank-1 penalties, which sharpen the distribution via the squeeze effect, as shown in the upper panel of Figure 4-(b).

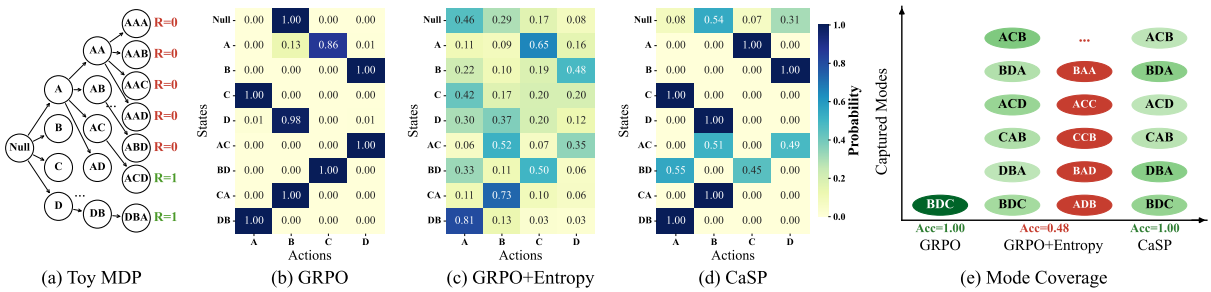
CaSP instead applies stronger negative updates only when the sampled token is the current rank-1 candidate. For tokens with $A_i < 0$, we replace the original ratio with

$$\gamma_{i,t}^{-} = \begin{cases} \lambda_{\text{top1}}\gamma_{i,t}, & y_{i,t} = v_{i,t}^{(1)}, \\ \gamma_{i,t}, & y_{i,t} \neq v_{i,t}^{(1)}, \end{cases} \quad (5)$$

where $\lambda_{\text{top1}} > 1$ controls the strength of the rank-1 penalty. This penalizes overconfident incorrect predictions strongly while avoiding the squeeze effect on lower-ranked candidates, as shown in the lower panel of Figure 4-(b).

4.3 A Didactic Toy Example

Setup. LLM generation under RLVR is naturally formulated as a finite-horizon, tree-structured MDP (He et al., 2025b). We study the learning dynamics in a tabular instance, where the policy samples actions from a categorical distribution at each state and a programmatic verifier provides reward at terminal states. This controlled setting aims to interpret how each method changes exploration. The full configuration is in the caption of Figure 5.



(a) Toy MDP (b) GRPO (c) GRPO+Entropy (d) CaSP (e) Mode Coverage

Figure 5: **Toy example: mode collapse in GRPO, accuracy degradation in GRPO+Entropy, and diversity preservation in CaSP.** (a) Tree MDP with horizon 3 and action space {A, B, C, D}; rewarding sequences: ACD, ACB, BDC, BDA, CAB, DBA. (b) GRPO collapses to a single mode. (c) GRPO+Entropy recovers all modes but leaks mass to non-rewarding actions (red borders), accuracy = 0.48. (d) CaSP preserves all modes without leaking to dead-end actions, accuracy = 1.00. (e) Mode coverage and accuracy over 1,000 responses.

Results. GRPO collapses to a single mode (BDC), placing nearly all root probability on action B. **GRPO+Entropy** recovers all six modes by flattening the distribution, but its entropy bonus indiscriminately spreads mass to non-rewarding actions at intermediate states (red borders in panel c), and accuracy degrades to 0.48. **CaSP** preserves all six modes while keeping accuracy at 1.00, since top-N redistribution confines exploration to plausible continuations rather than the full action set.

Discussion. The desideratum from Section 3 asks for non-negligible probability mass on the top-N candidates, and CaSP addresses this at the gradient level. Positive updates are spread among the top-N candidates rather than concentrated on the sampled token, and a stronger penalty on the rank-1 candidate prevents the squeeze effect from pushing mass back. Panel (d) verifies this, with every rewarding root action retaining support. Vanilla entropy regularization cannot match this behavior, since an entropy bonus rewards any spread of mass regardless of which candidates receive it and therefore leaks probability onto dead-end actions (red borders in panel c). As shown in Figure 4-(c), two distributions with identical entropy can differ sharply, so entropy alone cannot distinguish them. CaSP avoids this by acting on the identity of the top-N candidates rather than on a scalar proxy.

5 Experiments

5.1 Setup

Models and Datasets. For math tasks, we experiment with a diverse set of models, including Qwen2.5-32B (Yang et al., 2024a), Qwen2.5-Math-7B (Yang et al., 2024b), and Llama3.2-3B-Instruct (Dubey et al., 2024). The Qwen models are trained on the MATH (Level 3-5) dataset (Hendrycks et al., 2021), while the Llama model is trained on a combined dataset of GSM8K (Cobbe et al., 2021) and MATH (Level 1). For logical reasoning tasks, we train Qwen2.5-7B (Yang et al., 2024a) on the Synlogic-easy dataset (Liu et al., 2025a). For code reasoning, we train DeepSeek-R1-Distill-Qwen-7B on DeepCoder training dataset (Luo et al., 2025).

Evaluation Protocol. We compare CaSP against several competitive baselines, including GRPO, PSR, NSR, with entropy loss, W-REINFORCE (Zhu et al., 2025), KL-Cov (Cui et al., 2025), with “forking" tokens (Wang et al., 2025), P@k T. (Chen et al., 2025b), and Entropy-Adv (Cheng et al., 2025). Evaluations are conducted on a variety of reasoning benchmarks: MATH500 (Hendrycks et al., 2021), Minerva_math (Lewkowycz et al., 2022), Olympiad-Bench (He et al., 2024), AMC, AIME, Synlogic-easy (validation split) (Liu et al., 2025a), BBH (Suzgun et al., 2022), LiveCodeBench (Jain et al., 2025), and HumanEval+ (Liu et al., 2023). See more training and evaluation details in Appendix E.1 and E.2.

5.2 Math Reasoning

We evaluate CaSP on 6 math benchmarks across 3 backbones in Table 1. Compared to the base models, CaSP significantly improves the pass@1 score by 17.6% on Qwen2.5-Math-7B and 9.8% on Llama3.2-3B-Instruct. At the same time, it also boosts the pass@256 score by 4.1% and 1.9% on the same backbones respectively, demonstrating improved exploration and overall reasoning quality. Notably, on the larger Qwen2.5-32B model, CaSP further improves pass@1 by 19.6% and pass@128 by 0.6%, suggesting that CaSP remains effective when scaled to larger backbones.

Compared to GRPO and other baselines (KL-Cov, Entropy-Adv, w/ “forking” tokens and P@k

Method	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg.
Qwen2.5-Math-7B							
Base Model	13.2 / 66.0	5.4 / 51.8	38.2 / 98.5	55.8 / 96.0	16.5 / 68.8	25.6 / 77.0	25.8 / 76.4
GRPO	28.1 / 72.3	11.5 / 52.1	61.2 / 97.1	76.6 / 96.2	33.4 / 64.0	39.1 / 74.7	41.7 / 76.1
PSR	19.3 / 68.5	11.2 / 48.9	62.1 / 94.9	74.0 / 91.4	32.8 / 63.6	37.6 / 67.7	39.5 / 72.5
NSR	22.8 / 80.3	9.7 / 61.2	59.4 / 100.0	74.6 / 97.0	32.9 / 65.1	37.8 / 78.4	39.5 / 80.3
W-REINFORCE	29.2 / 86.5	10.8 / 55.7	61.1 / 97.4	76.4 / 96.4	33.4 / 67.6	38.1 / 77.6	41.5 / 80.2
KL-Cov	30.9 / 81.2	11.7 / 55.2	62.2 / 97.4	76.5 / 97.0	34.4 / 66.2	39.2 / 76.9	42.5 / 79.0
P@k T.	26.7 / 77.9	10.2 / 61.3	58.8 / 97.5	73.3 / 96.8	33.2 / 68.8	36.6 / 78.2	39.8 / 80.1
GRPO w/ Entropy loss	26.3 / 72.2	9.1 / 55.1	56.4 / 99.6	75.1 / 97.0	34.9 / 68.0	37.0 / 76.9	39.8 / 78.1
GRPO w/ Entropy-Adv	29.1 / 81.7	10.9 / 55.0	62.5 / 92.1	77.1 / 95.0	33.5 / 60.7	39.7 / 71.9	42.1 / 76.1
GRPO w/ forking tokens	28.6 / 74.6	11.5 / 57.4	59.6 / 96.7	77.4 / 94.4	33.9 / 65.1	39.6 / 72.4	41.8 / 76.8
CaSP	32.8 / 78.0	12.9 / 64.6	62.4 / 97.5	77.6 / 96.8	35.0 / 68.4	39.8 / 77.8	**43.4 / 80.5**
Δ(CaSP-GRPO)	+4.7 / +5.7	+1.4 / +12.5	+1.2 / +0.4	+1.0 / +0.6	+1.6 / +4.4	+0.7 / +3.1	+1.7 / +4.4
Qwen2.5-32B							
Base Model	9.7 / 63.3	4.9 / 50.0	42.1 / 100.0	64.5 / 97.0	27.0 / 67.6	28.1 / 77.3	29.4 / 75.9
GRPO	27.3 / 66.7	16.1 / 43.3	65.1 / 95.0	82.7 / 93.8	43.1 / 61.4	44.5 / 69.3	46.5 / 71.6
GRPO w/ Entropy loss	25.3 / 66.7	15.2 / 36.7	62.0 / 95.0	82.7 / 95.2	42.7 / 60.3	44.8 / 68.4	45.4 / 70.4
CaSP ($\alpha = 0.01$)	29.9 / 73.3	17.8 / 50.0	67.5 / 97.5	83.3 / 97.0	43.9 / 65.8	46.2 / 73.6	48.1 / 76.2
CaSP ($\alpha = 0.05$)	31.8 / 70.0	17.8 / 53.3	68.5 / 97.5	83.8 / 96.8	45.2 / 67.3	47.1 / 74.2	**49.0 / 76.5**
Δ(CaSP-GRPO)	+4.5 / +3.3	+1.7 / +10	+3.4 / +2.5	+1.1 / +3.0	+2.1 / +5.9	+2.6 / +4.9	+2.5 / +4.9
Llama3.2-3B-Instruct							
Base Model	3.4 / 51.7	0.7 / 46.7	20.3 / 94.9	37.8 / 93.6	10.1 / 59.2	12.7 / 67.1	14.2 / 68.9
GRPO	12.7 / 55.1	1.1 / 44.1	32.5 / 96.7	53.1 / 91.6	17.3 / 62.5	20.1 / 67.0	23.3 / 69.5
PSR	7.8 / 57.4	1.0 / 35.1	27.2 / 98.8	50.3 / 91.0	18.5 / 61.0	18.9 / 63.7	20.6 / 67.8
NSR	11.1 / 53.7	1.5 / 47.4	30.3 / 94.6	53.3 / 94.0	19.0 / 60.3	20.0 / 68.0	22.5 / 69.7
W-REINFORCE	13.3 / 51.7	1.1 / 42.1	31.4 / 96.3	52.4 / 92.8	16.7 / 59.9	19.6 / 65.8	22.4 / 68.1
CaSP	13.8 / 54.6	1.0 / 45.4	35.2 / 98.8	54.6 / 93.4	18.5 / 63.2	21.0 / 69.6	**24.0 / 70.8**
Δ(CaSP-GRPO)	+1.1 / -0.5	-0.1 / +1.3	+2.7 / +2.1	+1.5 / +1.8	+1.2 / +0.7	+0.9 / +2.6	+0.7 / +1.3

Table 1: Pass@1/256 results for Qwen2.5-Math-7B and Llama3.2-3B-Instruct, and pass@1/128 results for Qwen2.5-32B on MATH500, AIME 2024/25, Minerva_math, OlympiadBench, and AMC23.

T.), CaSP consistently outperforms them across all backbones and K. More importantly, CaSP delivers these gains without sacrificing exploration (pass@256) and with even stronger exploitation (pass@1). Compared to GRPO, CaSP improves pass@256 by 4.4% on Qwen2.5-Math-7B and 1.3% on Llama3.2-3B-Instruct, and improves pass@128 by 4.9% on Qwen2.5-32B, while also achieving higher pass@1 across all three backbones. Token-level variants provide less consistent gains: KL-Cov slightly improves pass@1 over GRPO (42.5% vs. 41.7%) but remains below CaSP in pass@256 (79.0% vs. 80.5%), while Entropy-Adv (42.1% / 76.1%) and training with "forking" tokens (41.8% / 76.8%) do not improve pass@256 over GRPO. Vanilla entropy loss encourages indiscriminate spreading rather than preserving plausible branches. This limitation is evident in its lower pass@1 compared with GRPO on Qwen2.5-Math-7B (39.8% vs. 41.7%) and Qwen2.5-32B (45.4% vs. 46.5%).

For NSR and W-REINFORCE, strong pass@256 performance is maintained, but often at the expense of much lower pass@1. In contrast, CaSP achieves a better balance on most backbones. On Qwen2.5-Math-7B, CaSP reaches a slightly higher pass@256 score (80.5% vs. 80.3% for NSR and 80.2% for W-REINFORCE) while clearly outperforming both in pass@1 (43.4% vs. 39.5% and 41.5%). A similar trend is observed on Llama3.2-3B-Instruct, where CaSP improves both pass@256 (70.8% vs. 69.7% and 68.1%) and pass@1 (24.0% vs. 22.5% and 22.4%). These results support our hypothesis that alleviating probability over-concentration improves pass@K performance, indicating a better balance between exploitation and exploration.

5.3 Response-level diversity

To verify that preserving local candidate support translates into meaningful solution diversity, we further analyze responses on AIME24 and AIME25. For each problem, we sample 256 responses. Following (Zhu et al., 2026), we measure strategy

Method	Synlogic								BBH							
	1	2	4	8	16	32	64	128	1	2	4	8	16	32	64	128
Base Model	3.1	4.9	7.5	11.1	15.4	20.0	24.5	28.7	42.4	59.3	74.4	**84.6**	**89.9**	**92.4**	**93.6**	**94.2**
GRPO	34.0	36.7	39.1	41.1	43.1	45.0	47.0	48.6	54.4	62.8	69.8	75.6	79.3	82.5	84.9	86.8
PSR	27.3	29.3	31.7	34.3	36.9	39.4	41.6	43.6	54.9	62.7	68.8	73.4	76.8	79.4	81.4	82.8
W-REINFORCE	0.8	1.3	1.8	2.4	2.9	3.4	3.8	3.9	15.4	21.9	27.8	32.3	35.4	37.2	38.3	38.8
CaSP	**34.6**	**38.3**	**41.8**	**45.3**	**48.1**	**50.7**	**53.1**	**55.1**	**58.4**	**69.2**	**77.3**	83.0	86.8	89.3	91.0	92.1

Table 2: Pass@K results for Qwen2.5-7B on Synlogic and BBH Datasets.

Method	LiveCodeBench		HumanEval+	
	Avg@16	Pass@16	Avg@16	Pass@16
Base Model	30.85	47.31	81.02	94.48
GRPO	33.92	53.41	80.25	94.48
CaSP	**36.02**	**54.48**	**83.90**	**96.32**

Table 3: Performance comparison on code reasoning benchmarks for DeepSeek-R1-Distill-Qwen-7B.

Method	Pass@1	Div. Score	Corr. Clust.	Oracle P@8
GRPO	19.8	1.57	4.03	41.99
KL-Cov	21.3	1.60	9.23	46.43
P@k T.	18.5	**1.86**	18.62	48.18
CaSP	**22.9**	1.83	**20.02**	**52.65**

Table 4: Response-level diversity on AIME24 and AIME25. Div. Score measures mathematical strategy diversity, while Corr. Clust. denotes the average number of distinct correct reasoning clusters. Oracle P@8 is computed over deduplicated reasoning clusters.

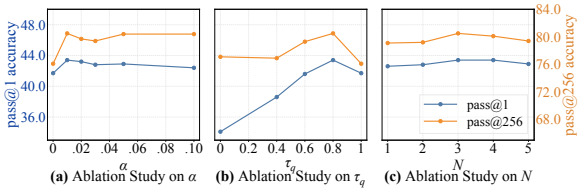


Figure 6: Ablations on α, τ_q, and N using Qwen2.5-Math-7B on math benchmarks.

diversity by randomly sampling 16 responses and asking GPT-5.5 to rate the diversity of their mathematical solution strategies on a 1–5 scale. We also cluster all 256 responses using Qwen3-Embedding-0.6B embeddings with average-linkage clustering; a cluster is considered correct if any response in the cluster reaches the correct answer. As shown in Table 4, CaSP achieves a strategy-diversity score of 1.83, comparable to the strongest baseline, Pass@K Training (1.86), while producing the most distinct successful trajectories (20.02 correct clusters), the highest deduplicated Oracle Pass@8 (52.65%), and the best pass@1 (22.9%). These results indicate that CaSP does not merely preserve local token alternatives, but converts candidate-level exploration into diverse reasoning trajectories.

5.4 Logical and Coding Reasoning

We evaluate the generalization of CaSP on two logic reasoning and two code reasoning benchmarks, as shown in Table 3 and Table 2.

On Synlogic, CaSP significantly outperforms the base model, with a +31.5% gain in pass@1 and +26.4% at pass@128. GRPO and PSR show improvements but lag behind CaSP by 6.5% and 11.5% at pass@128. W-REINFORCE, however, fails to train effectively, with pass@1 scores of only 0.8%. Similar observations hold on BBH.

On BBH, CaSP boosts the base model's pass@1 to 58.4% (+16.0%), and maintains stability at higher sampling rates, with a small 2.1% decrease in pass@128. GRPO and PSR, by comparison, drop 7.4% and 11.4% at pass@128 compared to the base model, showing difficulties in sustaining performance. W-REINFORCE performs poorly, achieving only 15.4% at pass@1. These results demonstrate that relying solely on negative samples is insufficient to improve pass@K on challenging tasks. In contrast, CaSP exhibits strong generalization, effectively trains on difficult tasks, and improves pass@K performance by mitigating probability over-concentration.

Table 3 presents the results on code reasoning benchmarks. On LiveCodeBench, CaSP achieves significant improvements, gaining +5.17% in pass@1 and +7.17% in pass@16, and outperforming GRPO by +2.1% and +1.07%, respectively. On HumanEval+, CaSP achieves the best performance with an Avg@16 score of 83.90%, while GRPO even degrades compared to the base model.

5.5 Ablations

We ablate the hyperparameters α, τ, and N, as well as the contributions of the positive and negative components. The full ablation results are in Appendix F.1, and the performance variations with respect to α, τ, and N are shown in Figure 6.

We evaluate α values from 0 to 0.1, with $\alpha = 0$ representing the performance of GRPO. Increasing α results in a monotonic improvement in pass@256, with gains ranging from 3.3% to 4.4%

Method	Time (s/step)	Memory (MiB/GPU)
GRPO	362.55±1.72	132,402
CaSP	366.44±1.64	132,458

Table 5: Training overhead on Qwen2.5-Math-7B using 8×H200 GPUs. Time is mean ± standard deviation over 10 measurements.

compared to GRPO. In contrast, pass@1 performance peaks at $\alpha = 0.01$ and then slightly degrades, though it remains superior to GRPO.

For τ, we sweep from τ_0 (all tokens receive CaSP updates) to τ_1 (no tokens, equivalent to GRPO). Notably, CaSP outperforms GRPO in pass@256 across most of τ_q. However, when CaSP is applied to all tokens, pass@1 drops by 9.3%. This indicates that CaSP should target high-entropy tokens instead of uniformly smoothing all.

For N, we test values from 1 to 5. As N increases, both pass@256 and pass@1 show an initial increase followed by a decrease. This trend suggests that restricting optimization to a small subset of the most probable candidates is sufficient. Specifically, pass@256 stays between 79.1% and 80.5%, while pass@1 fluctuates between 42.6% and 43.4%, both metrics outperforming GRPO.

5.6 Computational overhead.

CaSP introduces little additional training cost because it reuses the logits already computed for old-policy probability estimation and only extracts and stores the top-N candidate log-probabilities. As shown in Table 5, on Qwen2.5-Math-7B trained with 8×H200 GPUs, CaSP increases per-step training time by only 1.07% and increases peak GPU memory by less than 0.05%. Thus, candidate-level support preservation incurs negligible overhead relative to standard GRPO training.

6 Conclusion

We revisited exploration collapse in RLVR through the candidate-level distribution. Both positive and negative updates concentrate mass on the rank-1 candidate, collapsing rollout diversity regardless of K. To preserve mass on the top-N candidates, we proposed CaSP, which redistributes positive gradients across the top-N and applies a stronger penalty to the rank-1 candidate. CaSP improves pass@1 and pass@K simultaneously over GRPO and other strong baselines across math, logical, and coding benchmarks, with gains persisting up to 32B parameters and $K = 1024$.

Limitations

CaSP improves exploration by redistributing gradients within the local top-N candidate set, but this design also makes it dependent on the quality of the model's current candidate ranking. If useful reasoning branches are not assigned sufficient probability to enter the top-N set, CaSP cannot directly recover them. This limitation arises from a deliberate efficiency trade-off: operating on the full vocabulary would provide broader coverage but is computationally expensive and risks assigning probability mass to implausible tokens, while top-N redistribution keeps the update tractable and targeted. Extending CaSP to adaptively expand or refresh the candidate set is an interesting direction for future work.

Ethical Considerations

The study does not involve human subjects or the use of personal or sensitive data. All datasets and code utilized and released conform to their respective licenses and terms of use. The contributions in this work are foundational and do not raise issues related to fairness, privacy, security, or potential misuse. We confirm that all ethical considerations have been thoroughly addressed.

AI assistance. We used AI-assisted tools for language polishing, editing suggestions, and assistance with drafting code for internal analysis and plotting. All scientific claims, experimental results, analyses, citations, and final text were independently checked and finalized by the authors.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62606411) and the Westlake Education Foundation. The authors thank Westlake HPC Center for technical assistance.

References

Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. 2020. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446.

Xinzhu Chen, Xuesheng Li, Zhongxiang Sun, and Weijie Yu. 2025a. Beyond high-entropy exploration: Correctness-aware low-entropy segment-based advantage shaping for reasoning llms. *arXiv preprint arXiv:2512.00908*.

Zhipeng Chen, Xiaobo Qin, Youbin Wu, Yue Ling, Qinghao Ye, Wayne Xin Zhao, and Guang Shi. 2025b. Pass@ k training for adaptively balancing exploration and exploitation of large reasoning models. *arXiv preprint arXiv:2508.10751*.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*.

Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V Le, Sergey Levine, and Yi Ma. 2025. Sft memorizes, rl generalizes: A comparative study of foundation model post-training. *arXiv preprint arXiv:2501.17161*.

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

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. 2025. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*.

Runpeng Dai, Linfeng Song, Haolin Liu, Zhenwen Liang, Dian Yu, Haitao Mi, Zhaopeng Tu, Rui Liu, Tong Zheng, Hongtu Zhu, and Dong Yu. 2025. Cde: Curiosity-driven exploration for efficient reinforcement learning in large language models. *arXiv preprint arXiv:2509.09675*.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Wenlong Deng, Yi Ren, Muchen Li, Danica J Sutherland, Xiaoxiao Li, and Christos Thrampoulidis. 2025a. On the effect of negative gradient in group relative deep reinforcement optimization. *arXiv preprint arXiv:2505.18830*.

Wenlong Deng, Yi Ren, Yushu Li, Boying Gong, Danica J Sutherland, Xiaoxiao Li, and Christos Thrampoulidis. 2025b. Token hidden reward: Steering exploration-exploitation in group relative deep reinforcement learning. *arXiv preprint arXiv:2510.03669*.

Yihong Dong, Xue Jiang, Yongding Tao, Huanyu Liu, Kechi Zhang, Lili Mou, Rongyu Cao, Yingwei Ma, Jue Chen, Binhua Li, Zhi Jin, Fei Huang, Yongbin Li, and Ge Li. 2025. Rl-plus: Countering capability boundary collapse of llms in reinforcement learning with hybrid-policy optimization. *arXiv preprint arXiv:2508.00222*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Jingchu Gai, Guanning Zeng, Huaqing Zhang, and Aditi Raghunathan. 2025. Differential smoothing mitigates sharpening and improves llm reasoning. *arXiv preprint arXiv:2511.19942*.

Andre He, Daniel Fried, and Sean Welleck. 2025a. Rewarding the unlikely: Lifting grpo beyond distribution sharpening. *arXiv preprint arXiv:2506.02355*.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, et al. 2024. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems. *arXiv preprint arXiv:2402.14008*.

Haoran He, Yuxiao Ye, Qingpeng Cai, Chen Hu, Binxing Jiao, Daxin Jiang, and Ling Pan. 2025b. Random policy valuation is enough for llm reasoning with verifiable rewards. *arXiv preprint arXiv:2509.24981*.

Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. 2025c. Skywork open reasoner 1 technical report. *arXiv preprint arXiv:2505.22312*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Zhenyu Hou, Xin Lv, Rui Lu, Jiajie Zhang, Yujiang Li, Zijun Yao, Juanzi Li, Jie Tang, and Yuxiao Dong. 2025. Advancing language model reasoning through reinforcement learning and inference scaling. *arXiv preprint arXiv:2501.11651*.

Jian Hu. 2025. Reinforce++: A simple and efficient approach for aligning large language models. *arXiv preprint arXiv:2501.03262*.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. *Preprint*, arXiv:2503.24290.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Yuxian Jiang, Yafu Li, Guanxu Chen, Dongrui Liu, Yu Cheng, and Jing Shao. 2025. Rethinking entropy regularization in large reasoning models. *arXiv preprint arXiv:2509.25133*.

Katie Kang, Amrith Setlur, Dibya Ghosh, Jacob Steinhardt, Claire Tomlin, Sergey Levine, and Aviral Kumar. 2024. What do learning dynamics reveal about generalization in llm reasoning? *arXiv preprint arXiv:2411.07681*.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857.

Chenyi Li, Yuan Zhang, Bo Wang, Guoqing Ma, Wei Tang, Haoyang Huang, and Nan Duan. 2026. Setpo: Set-level policy optimization for diversity-preserving llm reasoning. *arXiv preprint arXiv:2602.01062*.

Jiazheng Li, Hong Lu, Kaiyue Wen, Zaiwen Yang, Jiaxuan Gao, Hongzhou Lin, Yi Wu, and Jingzhao Zhang. 2025a. Questa: Expanding reasoning capacity in llms via question augmentation. *arXiv preprint arXiv:2507.13266*.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. 2025b. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*.

Xiao Liang, Zhongzhi Li, Yeyun Gong, Yelong Shen, Ying Nian Wu, Zhijiang Guo, and Weizhu Chen. 2025. Beyond pass@ 1: Self-play with variational problem synthesis sustains rlvr. *arXiv preprint arXiv:2508.14029*.

Zihan Lin, Xiaohan Wang, Jie Cao, Jiajun Chai, Li Wang, Xiaodong Lu, Wei Lin, Ran He, and Guojun Yin. 2026. Resrl: Boosting llm reasoning via negative sample projection residual reinforcement learning. *arXiv preprint arXiv:2605.00380*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Junteng Liu, Yuanxiang Fan, Zhuo Jiang, Han Ding, Yongyi Hu, Chi Zhang, Yiqi Shi, Shitong Weng, Aili Chen, Shiqi Chen, et al. 2025a. Synlogic: Synthesizing verifiable reasoning data at scale for learning logical reasoning and beyond. *arXiv preprint arXiv:2505.19641*.

Yang Liu, Enxi Wang, Yufei Gao, Weixin Zhang, Bo Wang, Zhiyuan Zeng, Yikai Zhang, Yining Zheng, and Xipeng Qiu. 2026. The past is not past: Memory-enhanced dynamic reward shaping. *arXiv preprint arXiv:2604.11297*.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025b. Understanding r1-zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51. Notion Blog.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? *Advances in neural information processing systems*, 32.

Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. 2024. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*.

Noam Razin, Sadhika Malladi, Adithya Bhaskar, Danqi Chen, Sanjeev Arora, and Boris Hanin. 2025. Unintentional unalignment: Likelihood displacement in direct preference optimization. In *The Thirteenth International Conference on Learning Representations*.

Yi Ren. 2025. Learning dynamics of deep learning–force analysis of deep neural networks. *arXiv preprint arXiv:2509.19554*.

Yi Ren and Danica J. Sutherland. 2025. Learning dynamics of LLM finetuning. In *The Thirteenth International Conference on Learning Representations*.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

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

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

Yuda Song, Julia Kempe, and Remi Munos. 2025. Outcome-based exploration for llm reasoning. *arXiv preprint arXiv:2509.06941*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Remi Tachet, Mohammad Pezeshki, Samira Shabanian, Aaron Courville, and Yoshua Bengio. 2018. On the learning dynamics of deep neural networks. *arXiv preprint arXiv:1809.06848*.

Fahim Tajwar, Guanning Zeng, Yueer Zhou, Yuda Song, Daman Arora, Yiding Jiang, Jeff Schneider, Ruslan Salakhutdinov, Haiwen Feng, and Andrea Zanette. 2026. Maximum likelihood reinforcement learning. *arXiv preprint arXiv:2602.02710*.

Christian Walder and Deep Tejas Karkhanis. 2026. Pass@ k policy optimization: Solving harder reinforcement learning problems. *Advances in Neural Information Processing Systems*, 38:152416–152445.

Zhongwei Wan, Yun Shen, Zhihao Dou, Donghao Zhou, Yu Zhang, Xin Wang, Hui Shen, Jing Xiong, Chaofan Tao, Zixuan Zhong, Peizhou Huang, and Mi Zhang. 2026. DSDR: dual-scale diversity regularization for exploration in LLM reasoning. *CoRR*, abs/2602.19895.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, et al. 2025. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Fang Wu, Weihao Xuan, Ximing Lu, Zaid Harchaoui, and Yejin Choi. 2025. The invisible leash: Why rlvr may not escape its origin. *arXiv preprint arXiv:2507.14843*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024b. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Shihui Yang, Chengfeng Dou, Peidong Guo, Kai Lu, Qiang Ju, Fei Deng, and Rihui Xin. 2025a. DCPO: dynamic clipping policy optimization. *CoRR*, abs/2509.02333.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Yongxin Wang, Dongchun Xie, Yiwei Wang, Xiaodan Liang, and Jing Tang. 2025b. Depth-breadth synergy in rlvr: Unlocking llm reasoning gains with adaptive exploration. *arXiv preprint arXiv:2508.13755*.

Zhihe Yang, Xufang Luo, Zilong Wang, Dongqi Han, Zhiyuan He, Dongsheng Li, and Yunjian Xu. 2025c. Do not let low-probability tokens over-dominate in rl for llms. *arXiv preprint arXiv:2505.12929*.

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

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. In *Second Conference on Language Modeling*.

Kaichen Zhang, Shenghao Gao, Yuzhong Hong, Haipeng Sun, Junwei Bao, Hongfei Jiang, Yang Song, Hong Dingqian, and Hui Xiong. 2025. Rspo: Risk-seeking policy optimization for pass@ k and max@ k metrics in large language models. *arXiv preprint arXiv:2508.01174*.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. 2025. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. 2025. The surprising effectiveness of negative reinforcement in llm reasoning. *arXiv preprint arXiv:2506.01347*.

Xuekai Zhu, Daixuan Cheng, Dinghuai Zhang, Henry Li, Kaiyan Zhang, Che Jiang, Youbang Sun, Ermo Hua, Yuxin Zuo, Xingtai Lv, et al. 2026. Flowrl: Matching reward distributions for llm reasoning. In *International Conference on Learning Representations*, volume 2026, pages 151758–151780.

A More about the Design of $\gamma^{+}_{i,t}$

Derivation of Equation 3. For a fixed state $s_{i,t}$, let $\mathbf{z}_{i,t}$ be the logit vector and $\mathbf{e}_{y_{i,t}}$ be the one-hot target. At each decoding step, an LLM maps logits to the next-token distribution through a softmax layer. For any vocabulary token $v \in \mathcal{V}$, the softmax probability is

$$\pi_\theta(v \mid s_{i,t}) = \frac{\exp(z_v)}{\sum_{u\in\mathcal{V}} \exp(z_u)}. \tag{6}$$

Therefore, for the sampled token $y_{i,t}$,

$$\log \pi_\theta(y_{i,t} \mid s_{i,t}) = z_{y_{i,t}} - \log \sum_{u\in\mathcal{V}} \exp(z_u). \tag{7}$$

Taking the derivative with respect to each logit dimension z_v gives

$$\frac{\partial \log \pi_\theta(y_{i,t} \mid s_{i,t})}{\partial z_v} = \mathbf{1}[v = y_{i,t}] - \pi_\theta(v \mid s_{i,t}). \tag{8}$$

Equivalently, the sampled-token dimension has gradient

$$1 - \pi_\theta(y_{i,t} \mid s_{i,t}), \tag{9}$$

while every alternative token $v \neq y_{i,t}$ has gradient

$$-\pi_\theta(v \mid s_{i,t}). \tag{10}$$

Putting all dimensions together yields

$$\nabla_{\mathbf{z}_{i,t}} \log \pi_\theta(y_{i,t} \mid s_{i,t}) = \mathbf{e}_{y_{i,t}} - \pi_\theta(\cdot \mid s_{i,t}) := \mathbf{g}_{i,t}, \tag{11}$$

which gives Eq. 3.

This appendix provides more details about how we design Equation-(4.1):

$$\gamma^{+}_{i,t} = (1-\alpha)\,\gamma_{i,t} \;+\; \frac{\alpha}{N}\sum_{n=1}^{N} \mathrm{sg}\left(\frac{\gamma_{i,t}}{\gamma^{(n)}_{i,t}}\right)\gamma^{(n)}_{i,t},$$

Based on Eq. 4, the top-N smoothed target is

$$\tilde{\mathbf{e}}_{y_{i,t}} = (1-\alpha)\mathbf{e}_{y_{i,t}} + \frac{\alpha}{N}\sum_{n=1}^{N} \mathbf{e}_{v^{(n)}_{i,t}},$$

and its induced logit-gradient direction is

$$\begin{aligned}\tilde{\mathbf{g}}_{i,t} &= \tilde{\mathbf{e}}_{y_{i,t}} - \pi_\theta(\cdot \mid s_{i,t}) \\ &= (1-\alpha)\left(\mathbf{e}_{y_{i,t}} - \pi_\theta(\cdot \mid s_{i,t})\right) + \\ &\quad \frac{\alpha}{N}\sum_{n=1}^{N}\left(\mathbf{e}_{v^{(n)}_{i,t}} - \pi_\theta(\cdot \mid s_{i,t})\right) \\ &= (1-\alpha)\mathbf{g}_{i,t} + \frac{\alpha}{N}\sum_{n=1}^{N}\mathbf{g}^{(n)}_{i,t},\end{aligned} \tag{12}$$

where $\mathbf{g}^{(n)}_{i,t} := \nabla_{\mathbf{z}_{i,t}} \log \pi_\theta(v^{(n)}_{i,t} \mid s_{i,t})$ is the logit-gradient direction for the rank-n candidate.

Following the gradient-ratio decomposition used in Eq. 1, the unclipped GRPO term gives

$$\nabla_\theta\left(A_i\gamma_{i,t}\right) = A_i \cdot \mathrm{sg}(\gamma_{i,t}) \cdot \underbrace{\nabla_{\mathbf{z}_{i,t}} \log \pi_\theta(y_{i,t} \mid s_{i,t})}_{\mathbf{g}_{i,t}} \nabla_\theta \mathbf{z}_{i,t}\, \mathbf{g}_{i,t}.$$

Replacing $\mathbf{g}_{i,t}$ by the smoothed direction in Eq. 12 yields

$$\begin{aligned}A_i \cdot \mathrm{sg}(\gamma_{i,t}) \cdot \nabla_\theta \mathbf{z}_{i,t}\tilde{\mathbf{g}}_{i,t} &= (1-\alpha)A_i \cdot \mathrm{sg}(\gamma_{i,t}) \cdot \nabla_\theta \mathbf{z}_{i,t}\mathbf{g}_{i,t} \\ &\quad + \frac{\alpha}{N}\sum_{n=1}^{N} A_i \cdot \mathrm{sg}(\gamma_{i,t}) \cdot \nabla_\theta \mathbf{z}_{i,t}\mathbf{g}^{(n)}_{i,t}.\end{aligned} \tag{13}$$

The first term corresponds to the original sampled-token ratio $\gamma_{i,t}$. For the rank-n candidate, define

$$\gamma^{(n)}_{i,t} = \frac{\pi_\theta(v^{(n)}_{i,t} \mid s_{i,t})}{\pi_{\theta_{\text{old}}}(v^{(n)}_{i,t} \mid s_{i,t})}. \tag{14}$$

The naive mixture

$$(1-\alpha)\gamma_{i,t} + \frac{\alpha}{N}\sum_{n=1}^{N}\gamma^{(n)}_{i,t} \tag{15}$$

would use different importance coefficients for different rank-n candidates. We therefore multiply each auxiliary ratio by the stop-gradient correction $\mathrm{sg}(\gamma_{i,t}/\gamma^{(n)}_{i,t})$, so that its backward coefficient matches $\mathrm{sg}(\gamma_{i,t})$. This design is only one line of code, using the `.detach()` in `Pytorch`. This gives Eq. 4.1 and induces the smoothed gradient direction in Eq. 12.

B Proof for Section 3

B.1 Squeeze Effect of Negative Updates

Proof of Lemma 2. Fix a state $s_{i,t}$ and write $p_v = \pi_\theta(v \mid s_{i,t})$. Let z_v be the logit of token v, so that

$$p_v = \frac{\exp(z_v)}{\sum_{u\in\mathcal{V}} \exp(z_u)}. \tag{16}$$

For the sampled token $y_{i,t}$, its probability is

$$p_{y_{i,t}} = \frac{\exp(z_{y_{i,t}})}{\sum_{u\in\mathcal{V}} \exp(z_u)}. \tag{17}$$

For the sampled token itself, the softmax derivative is

$$\frac{\partial p_{y_{i,t}}}{\partial z_{y_{i,t}}} = p_{y_{i,t}}\left(1 - p_{y_{i,t}}\right). \tag{18}$$

For any alternative token $v \neq y_{i,t}$,

$$\frac{\partial p_{y_{i,t}}}{\partial z_v} = -p_{y_{i,t}} p_v, \qquad v \neq y_{i,t}. \tag{19}$$

For the unclipped GRPO term $A_i\gamma_{i,t}$, the denominator of $\gamma_{i,t}$ is fixed with respect to the current policy, so $\gamma_{i,t} \propto p_{y_{i,t}}$. When $A_i < 0$, gradient ascent on $A_i\gamma_{i,t}$ moves in the direction of decreasing $p_{y_{i,t}}$. For an alternative token v, the corresponding first-order logit increase is therefore proportional to $p_{y_{i,t}}p_v$:

$$\Delta z_v \propto p_{y_{i,t}}p_v, \qquad v \neq y_{i,t}. \tag{20}$$

Thus, among alternative tokens, the logit increase induced by the negative update is proportional to the current probability p_v up to the shared factor $p_{y_{i,t}}$. If $y_{i,t} \neq v_{i,t}^{(1)}$, where $v_{i,t}^{(1)}$ denotes the rank-1 candidate under $\pi_\theta(\cdot \mid s_{i,t})$, then $v_{i,t}^{(1)}$ has the largest probability among alternative tokens and receives the largest logit increase. □

B.2 Local Diversity under Concentration

Proof of Proposition 3. Let I_v indicate whether token v appears at least once among the K independent next-token samples $\tilde{y}_{1,t}, \ldots, \tilde{y}_{K,t}$:

$$I_v = \mathbf{1}\left[\exists k \in \{1, \ldots, K\} : \tilde{y}_{k,t} = v\right]. \tag{21}$$

The number of distinct sampled tokens is

$$|\{\tilde{y}_{1,t}, \ldots, \tilde{y}_{K,t}\}| = \sum_{v \in \mathcal{V}} I_v. \tag{22}$$

By linearity of expectation,

$$\mathbb{E}\left[|\{\tilde{y}_{1,t}, \ldots, \tilde{y}_{K,t}\}|\right] = \sum_{v \in \mathcal{V}} \Pr(I_v = 1). \tag{23}$$

Token v is absent from all K samples with probability $(1 - p_v)^K$, so

$$\Pr(I_v = 1) = 1 - (1 - p_v)^K. \tag{24}$$

Let $v^{(1)} \in \arg\max_{v \in \mathcal{V}} p_v$. The contribution of $v^{(1)}$ is $1 - (1 - p_{v^{(1)}})^K \to 1$ as $p_{v^{(1)}} \to 1$. For the remaining tokens, using $1 - (1 - a)^K \leq Ka$ for $a \in [0, 1]$,

$$\sum_{v \neq v^{(1)}} \left(1 - (1 - p_v)^K\right) \leq K \sum_{v \neq v^{(1)}} p_v = K(1 - p_{v^{(1)}}) \to 0. \tag{25}$$

Therefore, the expectation in Proposition 3 converges to 1 for fixed K. □

C A More Complete Related Work

C.1 Reinforcement Learning with Verifiable Rewards in LLMs

Reinforcement learning with verifiable rewards (RLVR) from large language models (LLMs) has demonstrated significant potential (DeepSeek-AI, 2025; Zeng et al., 2025; He et al., 2025c), especially when directly applied to a base model using GRPO (Shao et al., 2024) for RL training. This approach has notably enhanced the base model's performance, particularly in improving its reasoning abilities for mathematical and coding tasks. Subsequent works have focused on improving GRPO to further enhance the algorithm's performance. For instance, DAPO (Yu et al., 2025) adjusts GRPO's clipping thresholds and removes KL regularization to encourage larger updates in correct answer. Dr.GRPO (Liu et al., 2025b) eliminates the normalization term when computing advantages to prevent length bias. GSPO (Zheng et al., 2025) modifies the importance sampling from the token level to the sequence level, which proves to be more stable in training mixture-of-experts (MoE) models. Lopti (Yang et al., 2025c) identifies that low-probability tokens disproportionately influence GRPO training and accordingly reweights the advantage by token probability. These modifications have contributed to improvements in the model's pass@1 performance, but they have not specifically addressed pass@K performance, which relates to the model's exploration ability.

C.2 Effective Exploration for RLVR in LLMs

A central challenge in RLVR tasks lies in moving beyond the exploitation of a pretrained model's implicit knowledge to actively exploring diverse reasoning paths. Current methods tend to converge on a limited set of solutions, as evidenced by poor performance on the pass@K metric, which evaluates the coverage of multiple reasoning paths and thus reflects exploration effectiveness (Yue et al., 2025; Wu et al., 2025). To address this exploration deficit, existing methods can be roughly grouped by the level at which they intervene.

Response-level methods intervene at the level of whole trajectories. A representative example is pass@K training and related reward-shaping methods, which optimize the response distribution by directly reweighting sampled solutions (Walder and Karkhanis, 2026; Chen et al., 2025b; Li et al., 2026; Tajwar et al., 2026; Zhang et al., 2025; Li et al.,

2025b; Liu et al., 2026; Gai et al., 2025; Dai et al., 2025). These methods improve exploration in a trajectory-level sense, but they treat the model as a black box and do not explain which local candidates should receive probability mass.

Token-level methods further modify the local learning signal. Some methods use entropy as a proxy for exploration (Cui et al., 2025; Cheng et al., 2025; Wang et al., 2025; Hou et al., 2025; Hu et al., 2025; Jiang et al., 2025). Others use proxy signals such as confidence or hidden-state similarity to reweight token-level advantages (Deng et al., 2025b; Lin et al., 2026; Yang et al., 2025a). However, these proxies remain coarse summaries of exploration behavior and do not provide fine-grained control over which candidates should retain probability mass. A few recent methods combine token-level and response-level objectives (Yang et al., 2025a; Wan et al., 2026; Dai et al., 2025; Song et al., 2025).

Finally, *data-centric* methods expand the training distribution to expose the model to more diverse reasoning environments. One line of work uses off-policy data from more capable models to broaden the model's knowledge and promote solution diversity (Dong et al., 2025; Li et al., 2025a). Additional strategies include generating varied responses for challenging samples (Yang et al., 2025b) or paraphrasing questions to stimulate different reasoning trajectories for the same problem (Liang et al., 2025). These methods are effective to some extent, but they require either additional expert data or extra computation to generate diverse training data, rather than directly controlling how probability mass is allocated during training.

Another relevant line of work decomposes RLVR updates into positive and negative components, arguing that positive reinforcement sharpens the distribution while negative reinforcement enhances exploration (Zhu et al., 2025). However, as we analyze in Section 4.2, negative gradients do not uniformly promote exploration: when applied to low-probability tokens, they can in fact sharpen the distribution and induce a squeezing effect, which was not captured in prior analyses. Moreover, as shown in Sections 5.2 and 5.4, negative-only reinforcement improves exploration but at the cost of exploitation. It increases pass@K but consistently decreases pass@1. In particular, on tasks where the model must first consolidate exploitation before exploration becomes beneficial, strong negative reinforcement disrupts early pattern learning, ultimately reducing both pass@1 and pass@K. Our work addresses these limitations by providing a token-level, learning-dynamics–based perspective that explains these behaviors and motivates our proposed approach.

C.3 Analysis of Learning Dynamics in LLMs

Analyzing the learning dynamics of deep neural networks provides valuable insight into how training shapes model behavior (Saxe et al., 2013; Tachet et al., 2018; Advani et al., 2020). This analytical perspective has recently been extended to Large Language Models (LLMs), where prior work has widely examined the dynamics of supervised fine-tuning (SFT) (Kang et al., 2024; Chu et al., 2025), off-policy preference optimization methods such as DPO (Razin et al., 2025; Pal et al., 2024), or both (Ren and Sutherland, 2025; Ren, 2025).

Several recent studies have begun exploring the learning dynamics of on-policy RL. (Cui et al., 2025) adopt entropy-based metrics to track model changes during training. However, such metrics provide only an indirect signal by averaging over the entire vocabulary, thereby failing to capture meaningful shifts among high-probability candidates. In contrast, (Deng et al., 2025a) examine probability shifts induced by individual gradient updates to analyze inter-sample effects. While these analyses offer valuable fine-grained insights into probability changes, they fail to capture the cumulative evolution of the model's policy. To overcome these limitations, we propose a top-N probability dynamics framework that directly tracks how probability mass redistributes among the most likely candidates throughout training. This approach provides a scalable and interpretable lens for understanding how on-policy RL shapes model behavior.

D Additional Analysis of Distribution Changes During Training

D.1 Effects of CaSP on Training Dynamics

We analyze the training dynamics of CaSP in comparison with GRPO, KL-Cov, "forking" tokens, W-REINFORCE and Entropy-Adv. Figure 7 present the changes of top-N log-probabilities ($\Lambda^{(1)}$, $\Lambda^{(2)}$, and $\Lambda^{(3)}$) across training steps.

As can be seen, GRPO leads to severe over-concentration: $\Lambda^{(1)}_{\text{GRPO}}$ increases to nearly 0, while $\Lambda^{(2)}_{\text{GRPO}}$ and $\Lambda^{(3)}_{\text{GRPO}}$ sharply drop below -8 and -10, respectively. This indicates that nearly all probability mass collapses onto the top-1 token. KL-

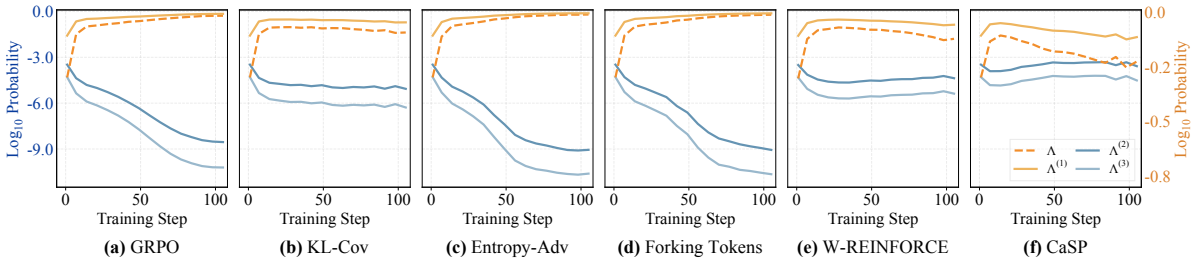


Figure 7: Comparison of CaSP with GRPO, KL-Cov, “forking" tokens, W-REINFORCE and Entropy-Adv on Qwen2.5-Math-7B. CaSP effectively controls probability concentration on the $\Lambda^{(1)}$ while preserving diversity among $\Lambda^{(2)}$ and $\Lambda^{(3)}$.

Cov exhibits a moderate concentration effect due to the KL penalty, while Entropy-Adv collapses even more rapidly, likely because of its stronger emphasis on high-entropy tokens. Training only on forking tokens further exacerbates probability over-concentration compared to GRPO, as it selectively increases the probability of high-entropy tokens. W-REINFORCE alleviates part of the over-concentration, but its sequence-level penalization of negative samples induces the squeezing effect in Section 4.2, leaving it still prone to probability concentration. In contrast, CaSP achieves the most effective deconcentration among all methods. This is evidenced by a lower $\Lambda^{(1)}_{\text{CaSP}}$ and higher $\Lambda^{(2)}_{\text{CaSP}}$ and $\Lambda^{(3)}_{\text{CaSP}}$. These results suggest that CaSP effectively mitigates probability mass collapse and can potentially encourages exploration during training.

To further validate this, we visualize the histogram of token-level entropy in Figure 8. GRPO drives most tokens toward near-zero entropy. CaSP, however, can preserve token entropy, particularly at “forking” points, where high entropy is desirable for exploration. This preservation of entropy further confirms CaSP’s role in promoting exploration.

D.2 Top-3 Candidates Probabilities Distribution

Figure 9 illustrates the probability-bin distributions of the top-3 candidates. The rank-1 candidate is concentrated in high-probability bins for all models, with the GRPO-trained model showing the strongest concentration. In contrast, the rank-2 and rank-3 candidates are mostly assigned to the lowest probability bin, and the rank-3 candidate is below 0.05 in more than 95% of cases across methods. This sharp decay from rank-1 to rank-3 indicates that most of the relevant probability mass is captured by the leading candidates, supporting our use of top-N candidate statistics as a tractable proxy for monitoring changes in the full next-token distribution.

E Implementation Details

E.1 Training Details

All experiments are conducted on 8 H100 GPUs. For the math tasks, training takes approximately 120 H100 GPU hours for the 7B models, while Qwen2.5-32B requires nearly 192 H200 GPU hours. For the logic tasks, training takes about 120 H100 GPU hours. For the coding tasks, training takes about 160 H100 GPU hours. We use verl v0.2.0 for training.

Math Tasks. For math tasks, all models are trained with a learning rate of 10^{-6}, a global batch size of 1024, and a PPO mini-batch size of 256. For each input problem, we sample 8 responses with temperature 1.0. The maximum prompt length is 1024. The maximum response length is 3072 for Qwen2.5-Math-7B and Llama3.2-3B-Instruct, and 8192 for Qwen2.5-32B. For the entropy-loss baseline, we set the coefficient to 0.01. For CaSP, we set $\alpha = 0.01$ and define the entropy threshold τ_q as the q-quantile of the token-level entropy distribution, such that a fraction q of tokens in each response have entropy lower than τ_q. Unless otherwise specified, we use $\tau_{0.8}$. We set $\lambda_{\text{top1}} = 1.1$ and $N = 3$ for all math models.

Logic Tasks. For logic tasks, all models are trained with a learning rate of 10^{-6}, a global batch size of 128, and a PPO mini-batch size of 32. For each input problem, we sample 16 responses with temperature 1.0. The maximum prompt length is 2048, and the maximum response length is 8192. We use dynamic sampling and set the upper clipping threshold to 0.28 for all methods. For CaSP, we apply a 50-step GRPO warm-up and set $\alpha = 0.005$, $\lambda_{\text{top1}} = 1.05$, and $N = 3$.

Coding Tasks. For coding tasks, all models are trained with a learning rate of 10^{-6}, a global batch

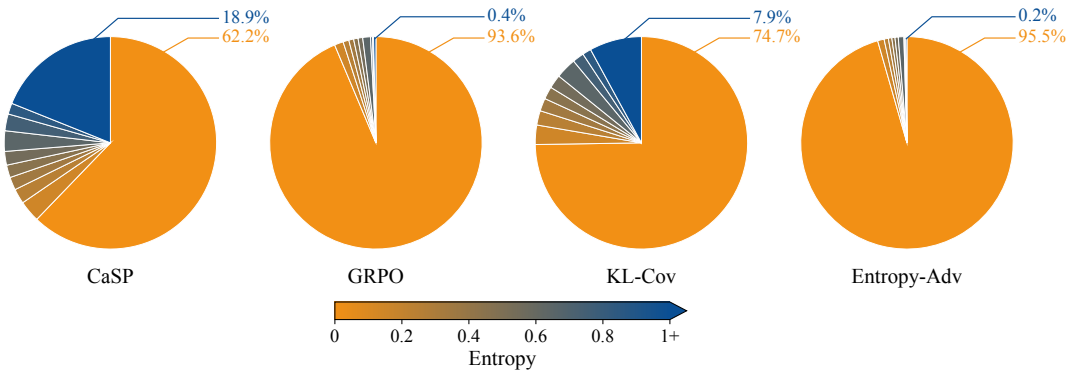


Figure 8: Token-level entropy distributions from the Qwen2.5-Math-7B backbone trained with CaSP, GRPO, KL-Cov, and Entropy-Adv, demonstrating CaSP's ability to maintain the entropy of the "forking" tokens.

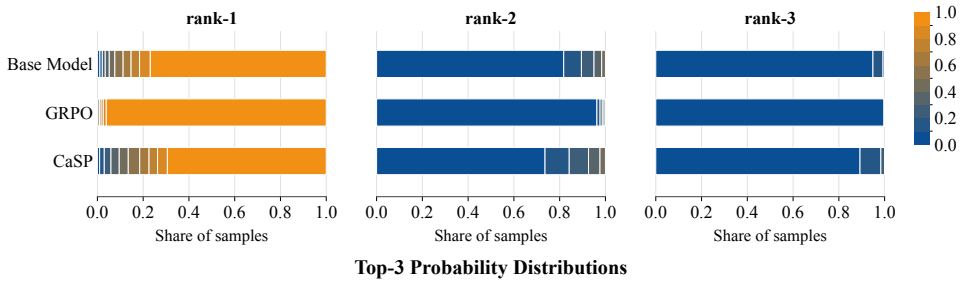


Figure 9: Probability-bin distributions for the top-3 next-token candidates. Across methods, the rank-1 candidate occupies substantially higher probability bins, whereas the rank-2 and rank-3 candidates are concentrated in the lowest bin. This pattern suggests that tracking the top-3 candidates captures the dominant changes in the model's next-token distribution.

size of 128, and a PPO mini-batch size of 32. For each input problem, we sample 8 responses with temperature 0.6. The maximum prompt length is 2048, and the maximum response length is 8192. For CaSP, we set $\alpha = 0.05$, $\lambda_{\text{top1}} = 1.1$, and $N = 3$.

E.2 Evaluation Details

Math Tasks. For math evaluation, we report pass@1 and pass@K. We use temperature 0.6 and top-$p = 0.95$ for sampling. The maximum generation length is 4096 for Qwen2.5-Math-7B and Llama3.2-3B-Instruct, and 8192 for Qwen2.5-32B. To reduce variance on small datasets such as AIME and AMC, we sample $n = 300$ responses per problem. For other math benchmarks, we use $n = 256$, except for Qwen2.5-32B where we use $n = 128$.

Logic Tasks. For logic evaluation, we sample $n = 128$ responses per problem with temperature 0.7, top-$p = 0.95$, and a maximum generation length of 8192. We report pass@1 and pass@K using the same evaluation protocol across all compared methods.

Coding Tasks. For coding evaluation, we sample $n = 16$ responses per problem with temperature 0.7, top-$p = 0.95$, and a maximum generation length of 8192. We report pass@1 and pass@K on the coding benchmarks.

E.3 PseudoCode

We present the pseudocode in Figure 10.

E.4 Artifacts and Licenses

We use publicly available models, datasets, and software artifacts in this work. The pretrained models include Qwen2.5-Math-7B, Qwen2.5-32B, Llama3.2-3B-Instruct, and DeepSeek-R1-Distill-Qwen-7B, each subject to the license and terms released by its original provider. The training and evaluation datasets, including MATH, GSM8K, SynLogic, BBH, LiveCodeBench, HumanEval+, AIME, AMC, Minerva Math, and OlympiadBench, are used according to their respective public licenses or benchmark usage terms.

E.5 Artifact Use and Data Considerations

We use publicly available models and benchmarks only for their intended research and evaluation purposes. The datasets are public research benchmarks, and we do not collect new user data or include personally identifying information.

```
def compute_policy_loss():
    ratio = torch.exp( log_prob - old_log_prob )
+    # 1. Identify forking tokens
+    w = (entropy > percentile(entropy, τ))
+    # 2. Using top-N ratio
+    topn_ratio = torch.exp(topn_log_probs -
     old_topn_log_probs)
+    topn_ratio = ((ratio.detach() / topn_ratio.
     detach())*topn_ratio).sum(dim=-1)
+    ratio = torch.where(advantage > 0, (1-α*w)*
     ratio + (α*w/K)*topn_ratio, ratio)
+    # 3. Apply a strong penalty to top1 negative
      tokens
+    mask = (advantage < 0) & is_top1 & w
+    ratio[mask] *= λ
    pg_losses = -advantage * ratio
    # ...clip and compute loss
```

Figure 10: The pseudo-code of the policy loss computation with CaSP. CaSP only requires modifying a few lines from a standard policy gradient implementation.

F More Experiment Result

F.1 Detailed Ablation Results

In this section, we present more detailed ablation results. As shown in Table 6, applying CaSP exclusively to either correct or incorrect responses leads to a drop in pass@K performance. This highlights the importance of both CaSP designs, as applying both yields the best results.

F.2 The Influence of Sampling Temperature

We evaluate Qwen2.5-Math-7B and its variants trained with GRPO and CaSP under different decoding temperatures ranging from 0.2 to 1.0. The results are shown in Table 7. Across all temperatures, CaSP consistently outperforms GRPO on both pass@1 and pass@K.

Benefiting from the probability smoothing introduced on “forking" tokens, CaSP maintains strong pass@256 performance even at very low temperatures. For example, at temperature 0.2, CaSP achieves 77.2% in pass@256, compared to only 63.8% and 69.7% for Qwen2.5-Math-7B and GRPO, respectively. This indicates that CaSP enables the trained model to retain a broader solution paths. At higher temperatures, all methods become more exploratory, yet CaSP consistently preserves a clear advantage. These results demonstrate that our conclusions are not sensitive to a narrow temperature choice and that CaSP is robust to this decoding hyperparameter.

F.3 Effect of Non-Uniform Top-N Weight Allocation

To examine whether non-uniform weight allocation within the top-N set helps, we implemented a probability-proportional softmax variant, referred to as CaSP (softmax), in place of the uniform smoothing used in the main CaSP method. The results in Table 8 show that this probability-aware smoothing yields a slight improvement in pass@K,indicating that allocating mass among plausible alternatives is indeed beneficial; however, it also introduces a decrease in pass@1 compare to CaSP. We regard this as a promising avenue for further refinement. At the same time, the observation reinforces our core motivation: maintaining diversity at “forking” tokens improves pass@K.

F.4 Scaling Sample K to 1024

We evaluate up to pass@1024 on AIME24/25 following (Yue et al., 2025). The results in Table 9 show that even in cases where the base model surpasses GRPO (K = 1024), CaSP still delivers large improvements over the base model. These findings further indicate that our method enhances exploration and has strong potential for handling more challenging tasks.

G Qualitative Analysis of Local Candidate Support

To qualitatively examine next-token exploration, we visualize grouped candidate matrices for two response segments in Figures 11–16. Columns denote generated token positions. For each position, candidates are added in descending probability order until their cumulative mass reaches 0.95. Thus, fewer visible candidates indicate stronger local probability concentration. Darker cells indicate higher probability, and the highlighted path marks the sampled sequence.

The visualizations show how many local candidates are needed to cover 95% of the next-token probability mass at each sampled position. The base model retains a small but visible set of alternatives around the sampled trajectory. After GRPO training, many positions require only one candidate to reach the 0.95 threshold, indicating stronger probability concentration and fewer effective branches for sampling.

In contrast, CaSP keeps broader candidate support around the sampled path: several positions require additional plausible candidates to reach the 0.95 cumulative mass threshold. This suggests that CaSP preserves alternative reasoning branches while maintaining the main trajectory, supporting our claim that top-N candidate support improves exploration without disrupting exploitation.

Table 6: Ablations on α, τ, k, γ^+, and γ^-. Pass@1/Pass@256 scores are evaluated using Qwen2.5-Math-7B.

Method	AIME24	AIME25	AMC	MATH500	Minerva	Olympiad	Avg.
Base Model	13.2 / 66.0	5.4 / 51.8	38.2 / 98.5	55.8 / 96.0	16.5 / 68.8	25.6 / 77.0	25.8 / 76.4
GRPO	28.1 / 72.3	11.5 / 52.1	61.2 / 97.1	76.6 / 96.2	33.4 / 64.0	39.1 / 74.7	41.7 / 76.1
α=0.02	31.1 / 79.9	**13.3** / 58.9	62.6 / 99.3	**77.6 / 97.0**	35.2 / 66.2	39.4 / 76.9	43.2 / 79.7
α=0.03	30.9 / 72.7	12.4 / 64.6	62.4 / 97.5	77.2 / 97.6	34.9 / 66.9	38.9 / 76.9	42.8 / 79.4
α=0.05	30.8 / 78.7	12.2 / **67.8**	63.0 / 97.5	77.2 / 96.8	34.8 / 65.1	39.3 / 76.3	42.9 / 80.4
α=0.1	29.1 / 75.5	12.4 / 65.0	61.8 / **99.9**	77.2 / 96.8	35.2 / 67.6	38.9 / **77.8**	42.4 / 80.4
τ_0	10.7 / 74.1	6.2 / 54.0	51.7 / 96.7	70.5 / 95.8	32.2 / 67.3	33.5 / 74.5	34.1 / 77.1
$\tau_{0.4}$	20.8 / 70.9	7.2 / 57.5	58.3 / 94.6	74.9 / 95.2	33.8 / 68.0	36.4 / 75.0	38.6 / 76.9
$\tau_{0.6}$	27.5 / 77.6	10.0 / 56.0	62.2 / **99.9**	76.5 / 96.8	**35.3 / 68.8**	38.2 / 76.4	41.6 / 79.3
$K = 1$	29.5 / 83.7	12.2 / 55.1	62.1 / 97.1	77.1 / 96.8	35.1 / 65.8	39.5 / 75.9	42.6 / 79.1
$K = 2$	30.3 / 78.4	12.2 / 57.9	62.5 / 99.6	77.2 / 96.4	34.9 / 65.8	39.5 / 76.9	42.8 / 79.2
$K = 4$	32.8 / **84.6**	12.5 / 54.6	62.6 / 97.5	77.7 / 97.2	35.3 / 68.4	39.2 / 78.4	43.4 / 80.1
$K = 5$	31.3 / 78.5	11.7 / 58.0	62.2 / 99.6	77.5 / 96.2	35.7 / 68.0	39.0 / 76.3	42.9 / 79.4
w/o γ^-	31.5 / 80.7	11.5 / 57.9	**62.7** / 97.4	77.1 / 96.2	34.1 / 65.8	39.0 / 75.6	42.8 / 78.9
w/o γ^+	30.4 / 75.2	12.7 / 64.9	62.3 / 99.6	77.4 / 96.4	34.7 / 65.8	39.5 / 77.3	42.8 / 79.9
CaSP	**32.8** / 78.0	12.9 / 64.6	62.4 / 97.5	**77.6** / 96.8	35.0 / 68.4	**39.8 / 77.8**	**43.4 / 80.5**

Table 7: Pass@1 / Pass@256 results for Qwen2.5-Math-7B across different temperatures on MATH500, AIME 2024/25, Minerva_math, Olympiadbench, and AMC23 Datasets.

Method	Temp	AIME24	AIME25	AMC23	MATH500	Minerva	Olympiad	Avg.
Base Model	0.2	13.3 / 55.0	5.3 / 41.9	40.7 / 82.5	62.8 / 93.2	14.8 / 44.1	28.2 / 66.2	27.5 / 63.8
GRPO	0.2	27.0 / 62.7	11.8 / 53.7	64.0 / 89.6	76.3 / 90.0	34.1 / 55.9	38.6 / 66.5	42.0 / 69.7
CaSP	0.2	**31.9 / 72.43**	**13.2 / 57.5**	**63.9 / 99.8**	**78.5 / 96.8**	**36.2 / 62.1**	**40.7 / 74.7**	**44.1 / 77.2**
Base Model	0.4	13.8 / 58.0	5.5 / 50.9	42.4 / 91.7	61.1 / 95.4	15.6 / 61.8	27.6 / 73.5	27.7 / 71.9
GRPO	0.4	26.7 / **75.6**	12.2 / 54.6	**63.4** / 94.5	76.4 / 93.2	33.5 / 61.0	38.6 / 71.6	41.8 / 75.1
CaSP	0.4	**31.1** / 72.3	**13.0 / 64.1**	62.9 / **97.5**	**78.1 / 97.4**	**36.1 / 66.2**	**40.6 / 76.1**	**43.6 / 78.9**
Base Model	0.6	13.2 / 66.0	5.4 / 51.8	38.2 / 98.5	55.8 / 96.0	16.5 / **68.8**	25.6 / **77.0**	25.8 / 76.4
GRPO	0.6	26.5 / 75.1	12.3 / 52.3	62.9 / 98.9	76.5 / 93.8	33.9 / 63.6	38.5 / 73.8	41.8 / 76.3
CaSP	0.6	**30.5 / 84.0**	**12.5 / 58.5**	**63.4 / 99.9**	**77.6 / 96.8**	**36.3** / 66.9	**40.0** / 76.6	**43.4 / 80.5**
Base Model	0.8	26.2 / 75.5	4.2 / 48.5	34.8 / **99.9**	50.8 / 96.8	17.0 / **69.9**	23.1 / **79.6**	23.5 / 76.7
GRPO	0.8	26.6 / **81.7**	12.0 / **52.3**	61.9 / 97.4	76.5 / 95.2	33.5 / 64.0	38.5 / 74.5	41.5 / 77.5
CaSP	0.8	**30.1** / 78.9	**12.5 / 52.3**	**63.2** / 99.5	**77.0 / 97.2**	**35.5** / 67.6	**39.2** / 77.5	**42.9 / 78.8**
Base Model	1.0	9.0 / 66.0	3.3 / 52.6	28.8 / **99.6**	42.8 / **97.2**	15.3 / **71.7**	19.0 / **80.4**	19.7 / 77.9
GRPO	1.0	26.2 /75.5	**11.7** / 51.8	61.5 / 97.1	**76.4** / 95.6	33.5 / 63.2	**38.3** / 75.7	41.3 / 76.6
CaSP	1.0	**28.9 / 75.6**	11.0 / **64.5**	**61.9** / 97.4	76.0 / 96.6	**34.8** / 69.5	37.9 / 77.2	**41.8 / 80.1**

Table 8: Pass@1 / Pass@256 results for Qwen2.5-Math-7B with the probability-proportional softmax variant of CaSP across MATH500, AIME 2024/25, Minerva_math, Olympiadbench, and AMC23 Datasets.

Method	AIME24	AIME25	AMC	MATH500	Minerva	Olympiad	Avg.
Base Model	13.2 / 66.0	5.4 / 51.8	38.2 / 98.5	55.8 / 96.0	16.5 / 68.8	25.6 / 77.0	25.8 / 76.4
GRPO	28.1 / 72.3	11.5 / 52.1	61.2 / 97.1	76.6 / 96.2	33.4 / 64.0	39.1 / 74.7	41.7 / 76.1
CaSP (softmax)	29.8 / **81.6**	12.4 / 63.2	61.5 / **99.6**	77.3 / **97.2**	34.0 / 66.9	39.4 / **77.8**	42.4 / **81.1**
CaSP	**32.8** / 78.0	**12.9 / 64.6**	**62.4** / 97.5	**77.6** / 96.8	**35.0** / 68.4	**39.8 / 77.8**	**43.4** / 80.5

Table 9: Pass@1024 results for Qwen2.5-Math-7B on AIME 2024/25 Datasets.

Dataset	Method	1	2	4	8	16	32	64	128	256	512	1024
AIME24	Base Model	13.4	21.3	29.6	36.7	42.9	49.1	55.2	60.8	66.1	72.0	80.8
	GRPO	27.0	33.3	39.2	45.0	51.4	58.1	64.6	70.4	74.4	76.2	76.7
	CaSP	**30.3**	**39.4**	**46.3**	**51.9**	**57.1**	**61.6**	**66.0**	**71.1**	**76.5**	**81.2**	**85.6**
AIME25	Base Model	5.6	9.5	14.7	20.8	27.0	32.9	38.6	44.3	50.5	57.9	65.5
	GRPO	12.0	17.3	23.4	29.5	34.7	39.2	43.5	47.9	53.1	58.3	62.3
	CaSP	**12.9**	**19.2**	**25.4**	**30.9**	**35.8**	**40.4**	**44.7**	**49.5**	**55.6**	**62.5**	**68.9**

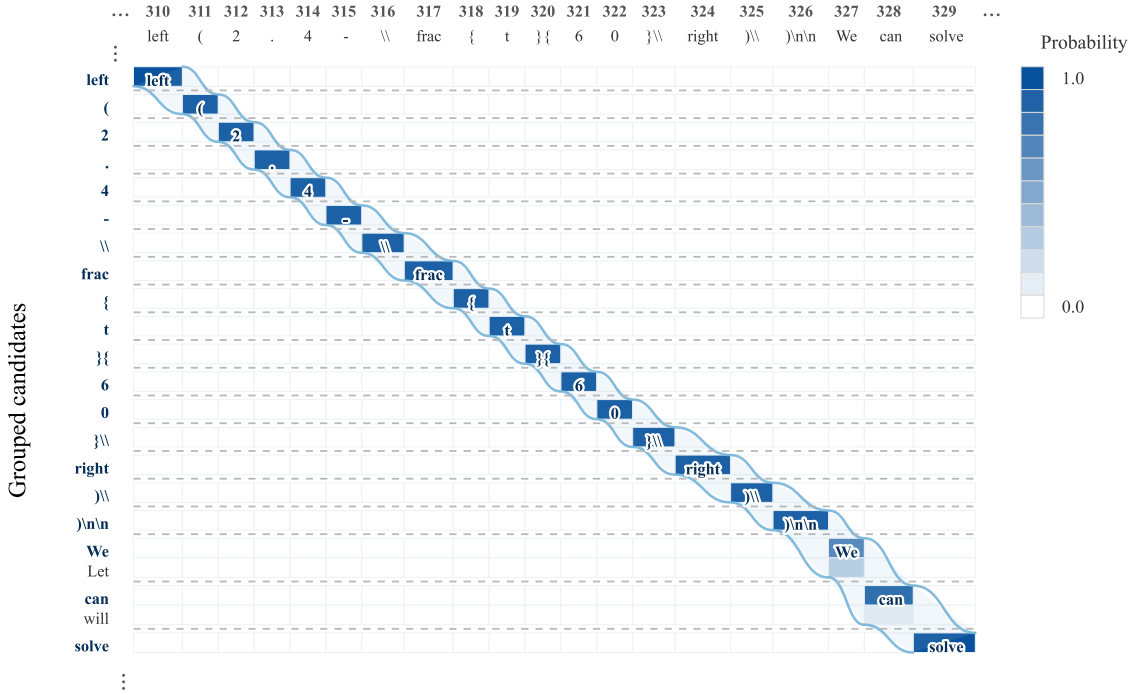


Figure 11: Grouped candidate matrix for the base model on response tokens 310–329. For each position, candidate tokens are shown in descending probability order until their cumulative mass reaches 0.95, with the sampled token always included. Fewer visible candidates therefore indicate stronger local probability concentration. Darker cells indicate higher next-token probability, and the highlighted path marks the sampled token sequence.

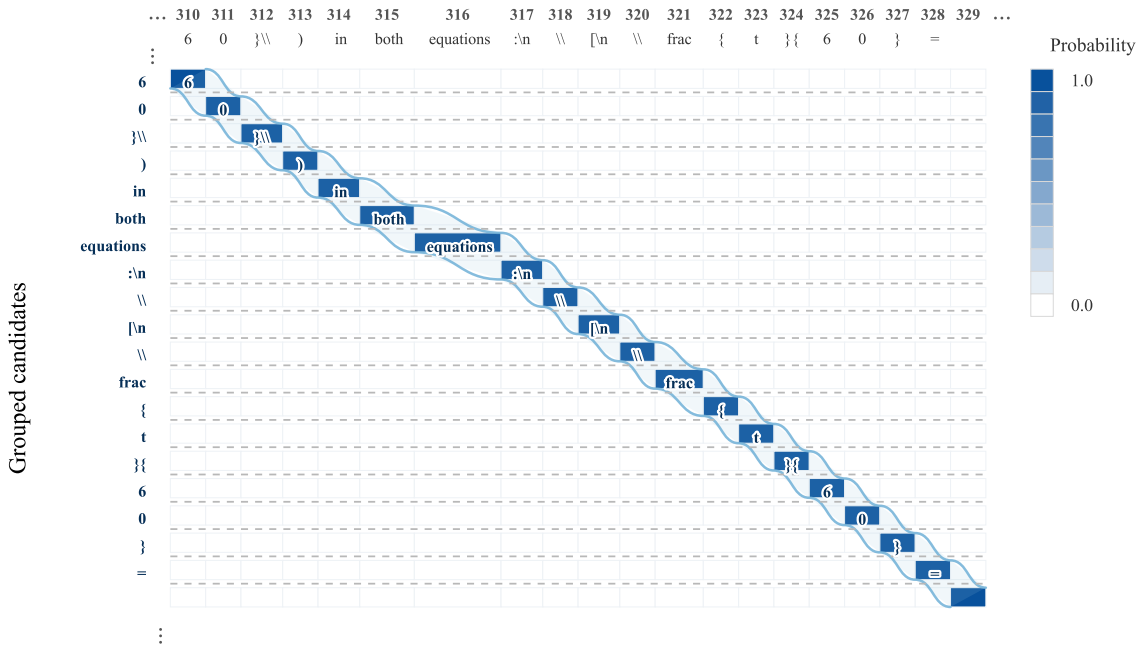


Figure 12: Grouped candidate matrix for the GRPO-trained model on response tokens 310–329. Compared with the base model, GRPO concentrates probability more strongly on the sampled/rank-1 path, leaving fewer visible high-probability alternatives.

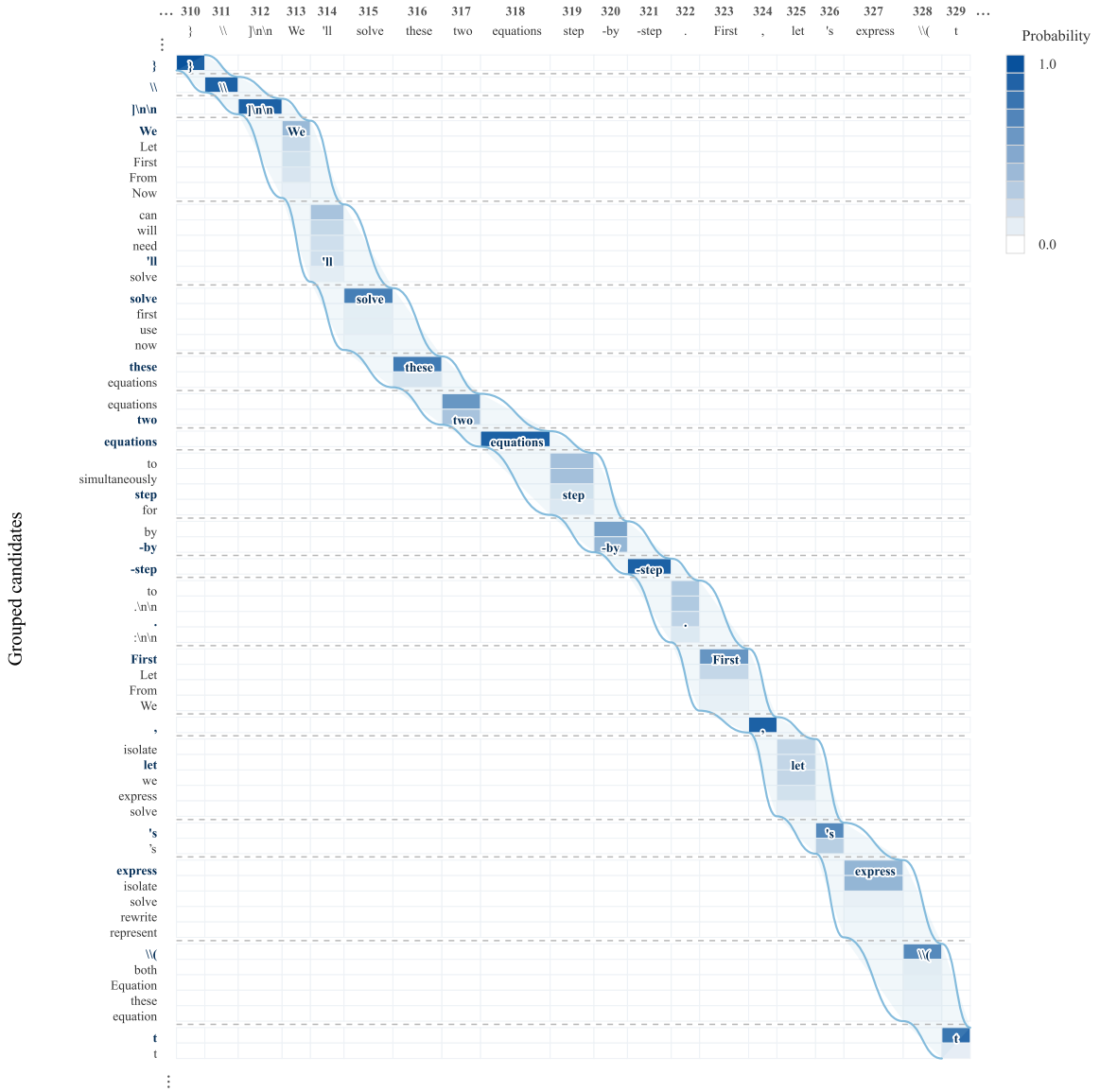


Figure 13: Grouped candidate matrix for the CaSP-trained model on response tokens 310–329. CaSP preserves a clearer sampled-token path while maintaining non-negligible probability on nearby candidate alternatives, indicating less severe local distribution collapse.

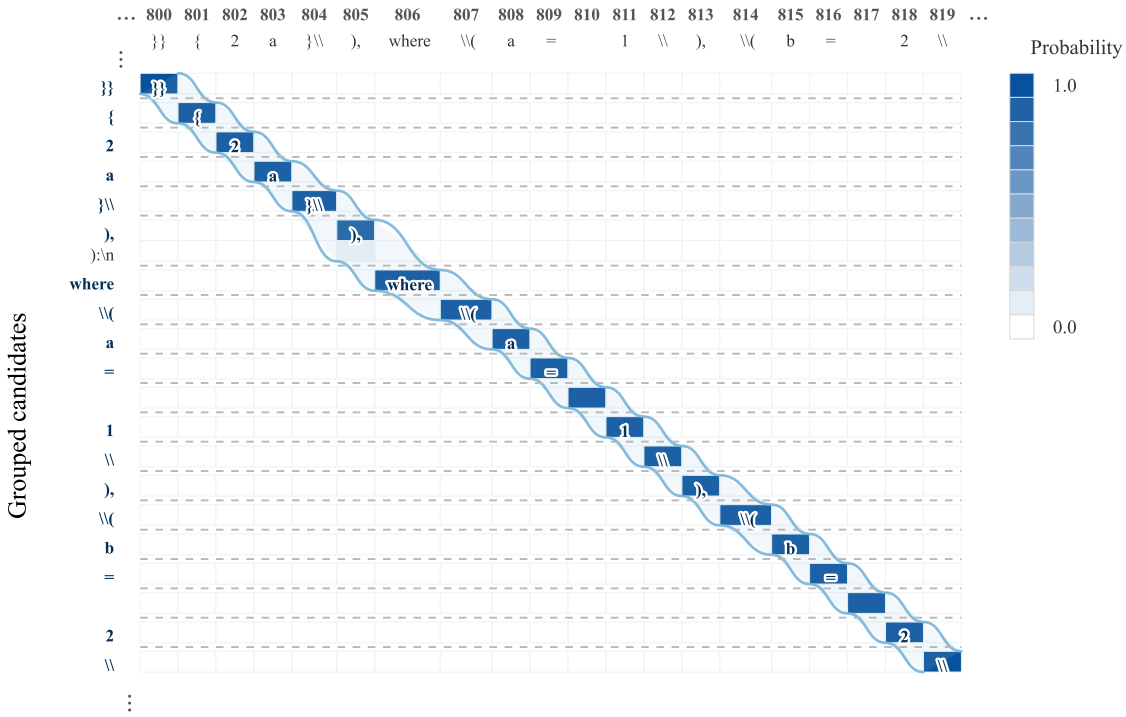


Figure 14: Grouped candidate matrix for the base model on response tokens 800–819.

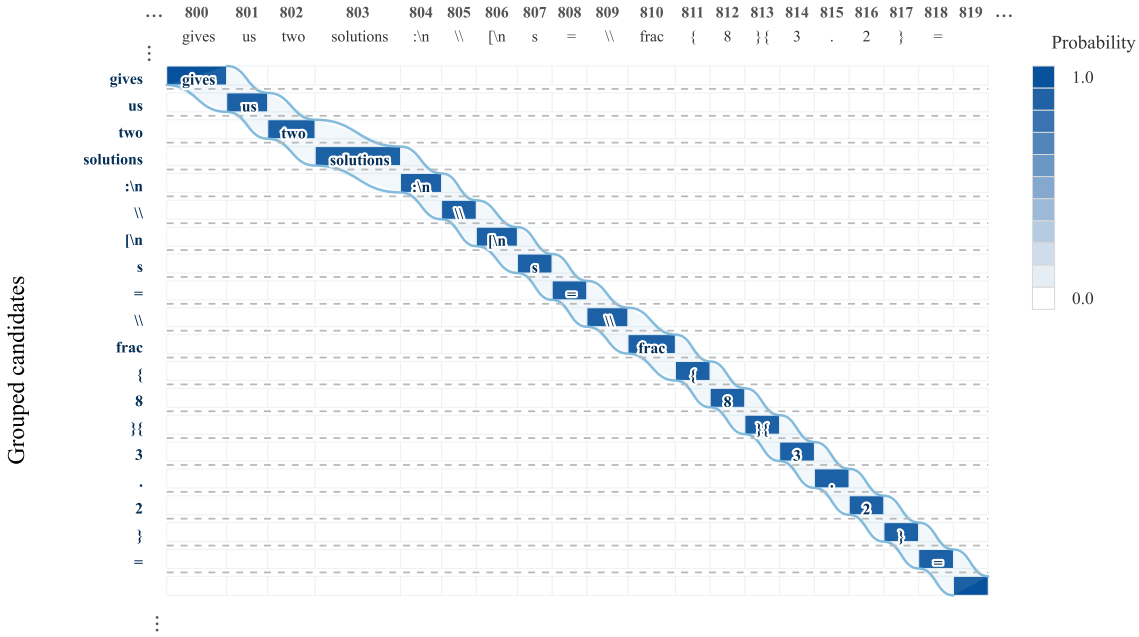


Figure 15: Grouped candidate matrix for the GRPO-trained model on response tokens 800–819. GRPO sharpens the local next-token distribution, concentrating probability on a narrow candidate path.

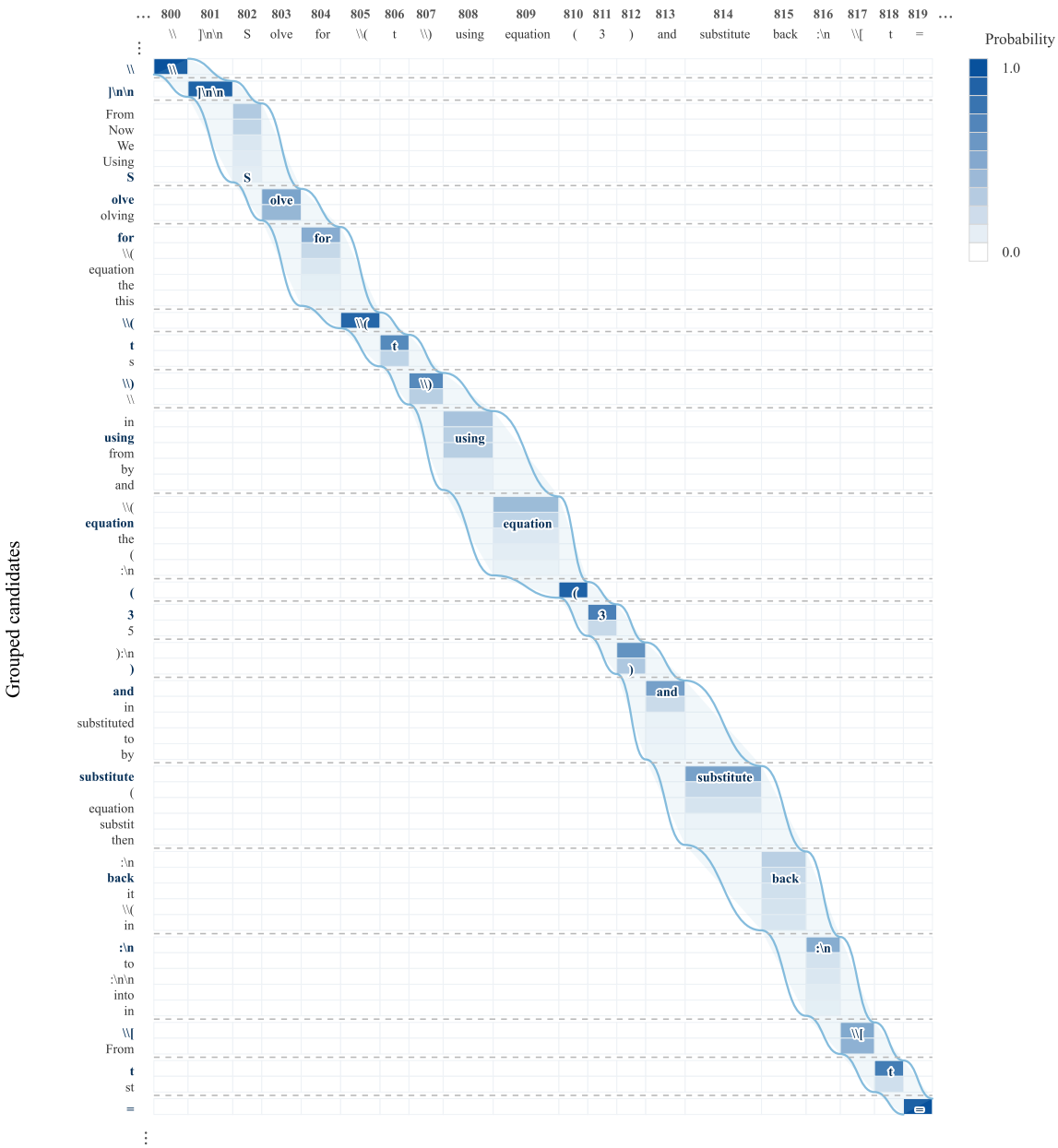


Figure 16: Grouped candidate matrix for the CaSP-trained model on response tokens 800–819. CaSP maintains broader local candidate support than GRPO while preserving the sampled reasoning trajectory.