DIVERSIFYING RLVR ROLLOUTS VIA FIRST-TOKEN EXPLORATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) trains reasoning models without labeled trajectories, using groups of verifier-scored rollouts to explore alternative reasoning paths. Limited rollout diversity is a central bottleneck, typically addressed through adjustments to temperature, prefixes, or rollout selection. We identify the *first token* of the response as a structurally distinct target for diversification, largely overlooked in prior work. We find that the first-token distribution is sharply concentrated and only weakly related to downstream correctness, as lower-probability candidates can yield similarly accurate responses. Diversifying the first token can therefore broaden the reasoning paths explored within each rollout group with little loss in response quality. Motivated by this observation, we introduce REFT (Rollout Exploration with First-Token Diversification), a lightweight modification to RLVR. REFT samples first tokens uniformly from the policy's top-N candidates and allocates rollouts evenly across the sampled tokens, leaving the rest of the pipeline unchanged. We evaluate REFT on eight models spanning multiple architectures and sizes (0.5B–14B), with mathematical reasoning and code-generation tasks under GRPO and DAPO. Across these settings, REFT consistently improves Pass@1, Pass@8, and Pass@64. It also outperforms competing diversification methods at every evaluated budget, incurring the lowest rollout cost.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become the standard recipe for training reasoning models (Guo et al., 2025; Hu et al., 2025). Instead of imitating annotated solutions, the policy samples candidate reasoning traces, an automatic verifier scores their final answers, and the model is updated from the resulting rewards. In grouped methods such as GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025), this update is constructed from a finite set of rollouts for the same prompt. The rollout group is therefore the learner's window onto the space of possible reasoning paths. If all rollouts receive the same reward, the group provides little contrastive signal. If the rollouts are semantically redundant, the verifier can only compare a narrow set of alternatives.

This makes rollout diversity a central bottleneck in RLVR. A prominent line of work looks for useful variation where the model already hesitates, at high-entropy reasoning pivots (Wang et al., 2025; Yu et al., 2026; Wei et al., 2026), trajectory-level branches (Xing et al., 2026; Hou et al., 2025; Zhao et al., 2026; Hu et al., 2026; Wan et al., 2026), or final-answer alternatives (Song et al., 2025; Wu et al., 2025; Zhu et al., 2025). Other methods broaden exploration through temperature-controlled sampling (Zhuang et al., 2025; Yang et al., 2025; Liu et al., 2025b), while efficiency-oriented methods reduce rollout cost through reuse and selective sampling (Huang & Wan, 2026; Liu et al., 2025a; Chang et al., 2026; Chen et al., 2026). These works differ in mechanism, but they largely leave low-entropy prefix positions comparatively overlooked as potential sources of useful variation.

We challenge this assumption by focusing on the *first token* of the response. In mathematical traces, this is usually a discourse opener such as "First", "Let", or "To", rather than a substantive reasoning step, so it appears too minor to matter. We show the opposite. The first token is low-load in semantic content, but high-leverage in distributional effect. Because it is the first autoregressive choice in the reasoning trace, changing it alters the conditional distribution over every subsequent token. Crucially, low probability at this position need not imply a poor chance of completing the problem correctly. Our diagnosis reveals a large mismatch between first-token probability and downstream correctness,

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.

where the first-token distribution is sharply biased, yet rollout correctness remains nearly flat across the top-20 alternatives on Qwen, Llama, and Gemma models from 3B to 72B. Moreover, this routing effect is not confined to the opener itself. After stripping the first token, continuations generated from different first tokens occupy different semantic regions, even though all subsequent tokens are sampled normally with the unchanged decoder, and uniform top-20 first-token sampling produces substantially higher diversity than standard sampling. Therefore, a sharply concentrated first-token prior can leave viable regions of the continuation space underexplored.

These observations motivate a different remedy to spend a fixed rollout budget, **REFT**: Rollout Exploration with First-Token Diversification. For each prompt, REFT takes the policy's own top-N valid first-token candidates, samples K of them uniformly, and allocates the rollout budget evenly across the selected first tokens. Beyond this first-token allocation, continuation decoding, verification, reward and advantage computation, the RL objective, and the total rollout budget are unchanged. REFT diversifies where the model is artificially certain and correctness is comparatively stable, rather than injecting noise where the model is uncertain and correctness is fragile.

Empirically, REFT improves Pass@1, Pass@8, and Pass@64 under both GRPO and DAPO on mathematical reasoning, across six models spanning 0.5B–14B parameters. The gains also extend to code-generation RLVR, and a matched comparison shows REFT outperforming prior diversification alternatives at every budget while using the lowest rollout cost tested. A first-token loss-masking control attributes the gains to broader rollout coverage rather than to corrected first-token crediting. Together, these results suggest a simple inversion of the usual exploration intuition. In RLVR, useful diversity can be recovered not only from high-entropy reasoning forks, but also from low-load first-token choices whose probability distribution is sharply biased yet only weakly tied to correctness.

2 Preliminaries

RLVR and grouped rollouts. Reinforcement learning with verifiable rewards (RLVR) trains a language-model policy from sampled attempts rather than labeled reasoning trajectories. Given a prompt x, the policy samples a *rollout* y, comprising reasoning trace and the final answer, and an automatic verifier returns a scalar reward $R(x, y)$. Group-based RLVR methods sample G rollouts for the same prompt and update the policy from their relative rewards. In GRPO (Shao et al., 2024), the reward for rollout y_i is converted to a group-normalized advantage

$$A_i = \frac{R_i - \text{mean}(\{R_j\}_{j=1}^{G})}{\text{std}(\{R_j\}_{j=1}^{G}) + \epsilon}, \tag{1}$$

where $R_i = R(x, y_i)$, and a PPO-style clipped objective (Schulman et al., 2017) then applies this trajectory-level advantage to the tokens of y_i. Therefore, the finite rollout group is the unit from which the learning signal is constructed and recent RLVR systems and variants build on this template. DeepSeek-R1 (Guo et al., 2025) scales verifier-based reasoning RL, and DAPO (Yu et al., 2025) improves GRPO with dynamic sampling, asymmetric clipping, token-level loss, and overlong-response handling. Recent methods further improve grouped-rollout data construction or usage. GRESO (Zheng et al., 2025a) filters uninformative prompts, Jackpot (Chen et al., 2026) budgets rejection sampling, AR3PO (Zhang et al., 2025) reuses rollouts, RL-ZVP (Le et al., 2026) learns from zero-variance prompts, PODS (Xu et al., 2025) selects informative rollout subsets for update, and saturated all-correct prompts are reheated to extract residual signal (Liu et al., 2025b).

Why rollout diversity matters. Equation (1) makes diversity a first-order concern. If all rollouts in a group receive the same reward, the group has no reward contrast and contributes little or no policy-gradient signal. Even when rewards are non-identical, a group of near-duplicate reasoning traces provides a narrow signal, because the verifier can only compare alternatives that were actually sampled. Diverse rollouts therefore serve two roles. They increase the probability that a finite group contains useful reward variation, and they broaden the set of reasoning regions over which the verifier can assign credit. Under Pass@k evaluation, retaining multiple plausible solution modes can matter as much as increasing the probability of the model response. Recent analyses of outcome-level diversity collapse (Song et al., 2025), support narrowing under RLVR (Wu et al., 2025), and sample-reinforcement dynamics (Zhu et al., 2025) point to a common concern that improving Pass@1 can narrow the accessible rollout distribution and hurt exploration or inference-time scaling.

Existing routes to rollout diversity. Maintaining rollout diversity has accordingly drawn a wide range of remedies. The most immediate is adjusting the sampling temperature, varied across rollouts, applied selectively to high-entropy tokens (Zhuang et al., 2025), or annealed from hot to cold along the sequence (Yang et al., 2025). Much recent work concentrates exploration where the model hesitates. Policy gradients are reserved for the high-entropy minority of "forking" tokens (Wang et al., 2025), ERPO amplifies advantages at high-entropy decision pivots (Yu et al., 2026), and rollout generation grows into trees, with Entropy-Tree and LATR branching where uncertainty runs high (Wei et al., 2026; Xing et al., 2026), FR3E relaunching from high-entropy decision points (Zheng et al., 2025b), and TreeRL searching on-policy (Hou et al., 2025). Others reward diversity directly, as an incentive in the objective (Hu et al., 2026; Wan et al., 2026), as bonuses over final answers (Song et al., 2025), or as a set-level Pass@k training target (Chen et al., 2025). Different as they are, these approaches share one instinct, how the response's first token is handled. It is treated as fixed scaffolding, amortized as a shared prefix (Huang & Wan, 2026; Chang et al., 2026), reused speculatively across iterations (Liu et al., 2025a), or pinned at temperature 1.0 by annealed decoding to preserve template-like text (Yang et al., 2025). Our question is complementary: can useful alternatives remain hidden at the very first choice that the model sees *no question to ask* and existing approaches largely *take for granted*?

3 Diagnosis: Low-Load, High-Leverage First Tokens

We define the *first token* as the earliest semantic token of the response, after leading whitespace and any control markers such as `<think>`. In math traces it is typically a discourse opener such as "First", "Let", or "To" (Appendix A.2) rather than an algebraic operation, a case distinction, or an answer. Given its limited semantic content, it may seem an unlikely place to find meaningful variation. This section shows that useful diversity can begin with this seemingly minor choice.

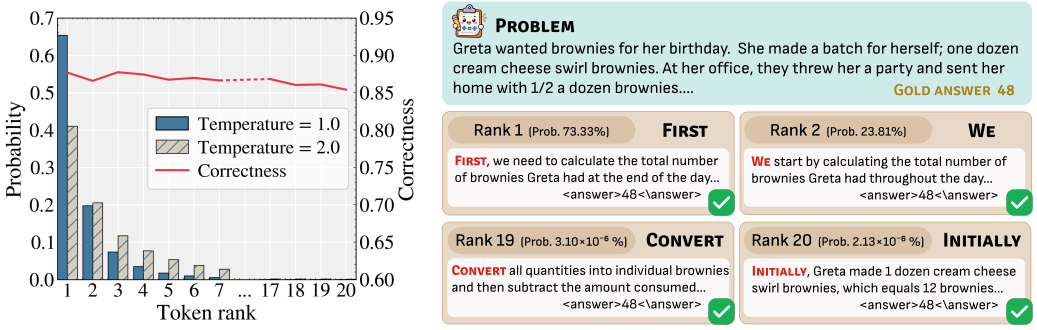


Figure 1: **Sharp probability, flat correctness.** (Left) The model assigns high probability to the top first token, but rollout correctness remains nearly flat across the top-20 ranks (Qwen2.5-7B-Instruct, GSM8K). (Right) Despite their low sampling probabilities, Convert (rank-19) and Initially (rank-20) produce natural continuations that reach the same correct answer as the rank-1 and rank-2 first tokens.

Low-probability first tokens perform reasonably well. For each prompt x, write $c(x)$ for everything the model conditions on before its first token, the prompt, template prefill, and any skipped markers (instantiated per model in Appendix C.5), and let $f_r(x)$ be the rank-r token of the first-token distribution $\pi_\theta(\cdot \mid c(x))$. The model's prior over this position is extremely sharp. Figure 1 (Left) shows that the top-ranked first token has mean probability 0.653 at temperature 1.0, versus 1.30×10^{-5} for rank 20. The surprise is how little correctness changes as probability falls so sharply across ranks. We fix the first token to be $f_r(x)$ and let the unchanged decoder complete the rollout. Correctness remains nearly flat across the top-20 ranks, with rank 20 achieving 85.41% against rank 1's 87.69%, despite a roughly $50{,}000$-fold difference in mean sampling probability (Figure 1 (Left); red line).

This is the central empirical asymmetry. At the first-token position, model probability and verifier correctness decouple. Standard sampling commits the group to a handful of openers before any substantive reasoning exists, yet a low-probability first token is not a bad reasoning choice. Up to 72B and recent models, the first-token distribution is strikingly concentrated despite only modest differences in correctness across alternative openings (Appendix A.1). Rank-1/rank-20 probability ratios span roughly 10^3–10^7, without a commensurate separation in downstream correctness.

The first token influences the whole continuation. The first token matters not only for what it says locally, but for how it conditions the rest of the rollout. Decomposing a response into the first token F and continuation z,

$$\pi_\theta(F, z \mid c(x)) \;=\; \pi_\theta(F \mid c(x))\, \pi_\theta(z \mid c(x), F), \qquad (2)$$

varying F induces distinct continuation distributions, and even a token with tiny marginal probability can open a competent but rarely visited region of solutions.

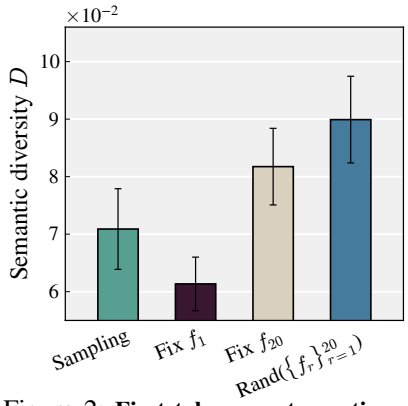


Figure 2: **First tokens route continuations.** Semantic diversity is measured after stripping the first token itself, so the effect comes from the rest of the rollout.

To test whether this routing effect is real, we compare four rollout strategies at matched group size: standard sampling, fixing $f_1(x)$, fixing $f_{20}(x)$, and sampling uniformly from the top-20 first tokens. We strip the first token from each rollout and measure the semantic diversity of what remains (Appendix D). Figure 2 shows that first-token choice changes the continuation distribution. Because diversity is measured after stripping, the effect is not surface variation in the first token. Even a fixed rank-20 opener yields more diverse continuations than standard sampling, and uniform sampling over the top-20 yields the most, suggesting that different first tokens contribute complementary rather than redundant variety. Comparable correctness and distinct continuations together mean the sharp prior is hiding *coverage*, solutions the group would benefit from contrasting yet standard sampling almost never visits.

Training can further sharpen the first-token prior. The bias is not only present before RLVR training; GRPO-style training can amplify it. The trajectory-level advantage is applied to every token in the rollout. For a rollout starting with F_i, the first-token gradient term is proportional to

$$A_i \nabla_\theta \log \pi_\theta(F_i \mid c(x)),$$

where A_i is the group-normalized advantage from Equation (1). This assigns the same scalar credit to the generic opener as to the later tokens that actually solve the problem. Because the top-ranked first token is sampled far more often, it also has far more chances to co-occur with positive-advantage trajectories. As Figure 3 shows, the top-1 first-token probability climbs steadily during training. The first-token sharpening we observe is consistent with prior work identifying entropy collapse as a barrier to exploration in RLVR (Yu et al., 2025; Zheng et al., 2025b; Liu et al., 2025c; Cui et al., 2025). By further concentrating the first-token distribution, standard GRPO can restrict access to diverse continuations before substantive reasoning begins.

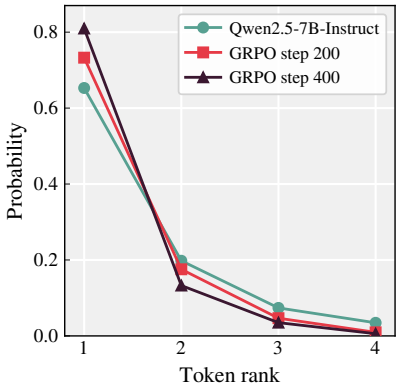
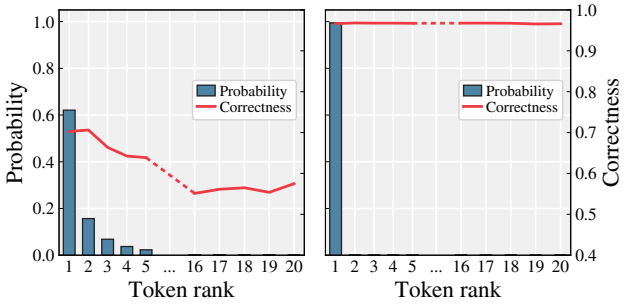


Figure 3: **RLVR sharpens the first-token prior.** The top-1 first-token probability increases during training, although the first token is only weakly tied to correctness.

The Gemma4-31B (Team et al., 2026) base–instruct comparison in Figure 4 provides an informative contrast. The base prior is substantially softer and its correctness declines across ranks, whereas the instruct prior collapses onto a single token even though correctness flattens. While this comparison cannot attribute the sharpening specifically to RLVR, it suggests that post-training can introduce or amplify a strong preference for particular first tokens without a commensurate separation in downstream correctness. Full base–instruct curves through rank 20 are in Appendix A.1 (Figure 8).



(a) Base (b) Instruct

Figure 4: **A sharper first-token prior after post-training.** Gemma4-31B on GSM8K. The instruct model concentrates on rank 1, unlike the relatively broader, rank-sensitive base model.

Why temperature is not the answer. A natural remedy for increasing rollout diversity is to raise the sampling temperature. However, the prior remains too sharp for temperature to fix at acceptable cost. As shown in Figure 1, even T=2.0 leaves the rank-20 candidate with probability $\sim 1.8 \times 10^{-3}$, and with group size G=8, the probability of sampling it at least once is $\sim 1.4\%$. Meanwhile, every later conditional along the trajectory, including fragile mid-reasoning steps, is flattened as collateral. So, what we need is not a global increase in sampling temperature, but targeted coverage of plausible first-token alternatives. We also compare REFT against temperature-based alternatives in Section 5.3.

4 REFT: Rollout Exploration with First-Token Diversification

Section 3 identifies the first token as a promising target for exploration. It is *low-load*: diversifying across lower-ranked first tokens incurs little loss in correctness. It is *high-leverage*: lower-probability candidates are rarely sampled, yet changing the first token can affect every subsequent conditional distribution. Together, these make the first token a surprisingly efficient exploration site, one that the sharp prior leaves underexplored under standard sampling.

These observations motivate **REFT** (Rollout Exploration with First-Token Diversification), which changes only first-token sampling. For each prompt, REFT samples first tokens uniformly from the policy's top candidates. This encourages each rollout group to explore a broader range of starting tokens rather than concentrate on the most probable ones. Subsequent tokens are generated as usual.

Candidate set. Given a prompt x, we compute the policy distribution at the first-token position and keep the top-N valid candidates:

$$\mathcal{F}_N(x) = \{f_1(x), \ldots, f_N(x)\}, \qquad \pi_{\theta_{\text{old}}}(f_1 \mid c(x)) \geq \cdots \geq \pi_{\theta_{\text{old}}}(f_N \mid c(x)).$$

Here $f_1, \ldots, f_N$ are the highest-probability valid first tokens under the behavior policy. Because the candidates come from the policy's own top-N set, REFT does not introduce external prompts or hand-written prefixes. It reallocates budget among first tokens the model already considers plausible.

First-token diversification. Standard sampling draws the first token of each of the G rollouts from the policy distribution. A simple alternative samples G first tokens uniformly from $\mathcal{F}_N(x)$. However, exploring many first tokens leaves few continuations per token, making each token's expected return difficult to estimate reliably. To balance coverage and repeated sampling, REFT selects K first tokens uniformly without replacement, with K dividing G:

$$\mathcal{S}_K(x) \sim \text{Unif}\left(\{\mathcal{S} \subseteq \mathcal{F}_N(x) : |\mathcal{S}| = K\}\right).$$

For each token in $\mathcal{S}_K(x)$, we generate G/K independent continuations using the standard sampling procedure, yielding G rollouts in total:

$$z_{f,j} \sim \pi_{\theta_{\text{old}}}\left(\cdot \mid c(x), f\right), \qquad j = 1, \ldots, G/K.$$

Smaller K allows more continuations per selected token, while $K = G$ gives one per token and recovers the simple alternative above. The resulting rollout group is

$$\mathcal{Y}_G^{\text{REFT}}(x) = \{(f, z_{f,j}) : f \in \mathcal{S}_K(x),\ j = 1, \ldots, G/K\}.$$

Therefore, REFT replaces sampling from a sharply peaked first-token distribution with explicit coverage of K plausible first tokens, while leaving all subsequent generation unchanged.

Algorithm 1 REFT rollout sampler

Require: Prompt x with response-start context $c(x)$, behavior policy $\pi_{\theta_{\text{old}}}$, group size G, candidate width N, selected-token count K
Ensure: Rollout group $\mathcal{Y}_G^{\text{REFT}}(x)$

$\mathcal{F}_N(x) \leftarrow \{f_1(x), \ldots, f_N(x)\}$ {Top-N valid tokens under $\pi_{\theta_{\text{old}}}(\cdot \mid c(x))$}
$\mathcal{S}_K(x) \sim \text{Unif}(\{\mathcal{S} \subseteq \mathcal{F}_N(x) : |\mathcal{S}| = K\})$
For each $f \in \mathcal{S}_K(x)$, sample $z_{f,1}, \ldots, z_{f,G/K} \sim \pi_{\theta_{\text{old}}}\left(\cdot \mid c(x), f\right)$
$\mathcal{Y}_G^{\text{REFT}}(x) \leftarrow \{(f, z_{f,j}) : f \in \mathcal{S}_K(x),\ j = 1, \ldots, G/K\}$
return $\mathcal{Y}_G^{\text{REFT}}(x)$

Table 1: **Mathematical reasoning across models and RLVR objectives.** GRPO and DAPO are compared with their REFT variants. Math Avg. excludes GSM8K, and **bold** marks the higher score.

	GSM8K		MATH-500		AIME24		AIME25		AMC23		Math Avg.	
Method	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@8
Qwen2.5-3B-Instruct	80.14	94.16	51.40	78.00	6.67	16.67	0.00	13.33	35.00	75.00	23.27	45.75
GRPO	84.53	95.38	59.80	79.40	10.00	20.00	3.33	16.67	42.50	77.50	28.91	48.39
GRPO + REFT	**86.28**	**96.13**	**61.00**	**81.20**	**13.33**	**23.33**	**6.67**	**20.00**	**47.50**	**85.00**	**32.13**	**52.38**
DAPO	86.05	95.15	60.00	78.60	10.00	20.00	6.67	16.67	45.00	80.00	30.42	48.82
DAPO + REFT	**86.20**	**96.06**	**61.20**	**80.00**	**13.33**	**26.67**	**13.33**	**20.00**	**52.50**	**85.00**	**35.09**	**52.92**
Llama3.2-3B-Instruct	73.24	91.96	29.80	64.00	3.33	13.33	0.00	3.33	20.00	50.00	13.28	32.66
GRPO	75.66	92.12	35.60	64.80	6.67	16.67	3.33	3.33	22.50	60.00	17.02	36.20
GRPO + REFT	**77.63**	**94.09**	**36.80**	**65.40**	6.67	**20.00**	3.33	**6.67**	**25.00**	**62.50**	**17.95**	**38.64**
DAPO	73.84	93.10	37.20	65.60	10.00	23.33	3.33	6.67	22.50	60.00	18.26	38.90
DAPO + REFT	**78.47**	**94.01**	**41.00**	**67.00**	**13.33**	23.33	**6.67**	**10.00**	**25.00**	**65.00**	**21.50**	**41.33**
Qwen2.5-7B-Instruct	88.48	96.74	53.40	80.00	10.00	20.00	0.00	23.33	37.50	75.00	25.23	49.58
GRPO	91.05	97.12	72.00	86.40	13.33	23.33	10.00	26.67	55.00	85.00	37.58	55.35
GRPO + REFT	**91.89**	**97.42**	**73.60**	**87.20**	**20.00**	**30.00**	**16.67**	**36.67**	**65.00**	85.00	**43.82**	**59.72**
DAPO	91.36	96.74	72.40	86.40	16.67	26.67	10.00	30.00	57.50	80.00	39.14	55.77
DAPO + REFT	**92.27**	**97.27**	**73.00**	**86.80**	**20.00**	**30.00**	**13.33**	**33.33**	**62.50**	**85.00**	**42.21**	**58.78**

What remains unchanged. After constructing $\mathcal{Y}_G^{\mathrm{REFT}}(x)$, optimization proceeds exactly as in the underlying RLVR algorithm. The verifier scores each rollout, group-normalized advantages are computed as usual, and the same GRPO/DAPO-style objective is applied to the sampled trajectories. REFT acts entirely within rollout construction, as a drop-in modification for the sampler's first decision, and composes with existing refinements such as dynamic sampling and asymmetric clipping.

Rollout construction with a fixed RL loss. REFT alters the data-collection distribution only at a single position, and the completed rollouts are trained with the same GRPO/DAPO objective as the baselines, without an importance correction for the first-token sampler. The resulting update therefore reflects REFT's reallocation of the finite rollout budget across plausible first tokens, rather than an unbiased estimate of standard on-policy sampling. Correcting back toward the original first-token prior would counteract the intended coverage. Further discussion is in Appendix E.

5 Experiments

5.1 Experimental Setup

Mathematical reasoning. We compare GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025) with their REFT-augmented variants. Training covers six models spanning 0.5B to 14B: Qwen2.5-3B/7B-Instruct (Yang et al., 2024) and Llama3.2-3B-Instruct (Grattafiori et al., 2024) on GSM8K (Cobbe et al., 2021), BigMath-Easy, and BigMath (Albalak et al., 2025); Qwen2.5-0.5B-Instruct on GSM8K, all under both objectives; Qwen3.5-9B (Team, 2026) and Qwen2.5-14B-Instruct on BigMath with GRPO. GSM8K uses group size G=8, and BigMath subsets use G=16. GSM8K-trained models are evaluated on GSM8K; BigMath-trained models on MATH-500 (Lightman et al., 2024), AIME 2024/2025 (Zhang & Math-AI, 2024; 2025), and AMC 2023 (Li et al., 2024). We report Pass@1, Pass@8, and Pass@64, with Math Avg. the unweighted average of the four non-GSM8K benchmarks.

Code generation. DeepSeek-R1-Distill-Qwen-7B (Guo et al., 2025) and Qwen2.5-Coder-3B-Instruct (Hui et al., 2024) are trained with GRPO (G=8) on a decontaminated, test-verified Python set (Open R1, 2025), using the fraction of passed sandbox unit tests as reward. The former generates an explicit `<think>` trace, whereas the latter generates Python directly. Evaluation reports Pass@1 and Pass@8 on HumanEval+, MBPP+ (Liu et al., 2023), and LiveCodeBench v4 (Jain et al., 2025).

Common configuration. Unless stated otherwise, REFT uses candidate width $N = 20$, selected-token count $K = 4$, allocating rollout slots evenly across selected first tokens. Candidate-width and replication ablations are in Appendix B.3. Within each baseline–REFT comparison, all methods share the same rollout budgets and identical decoding configurations. Training rollouts are generated with vLLM (Kwon et al., 2023). Full training and evaluation configurations are in Appendix C, and runtime measurements for the profiled configurations are reported in Appendix C.3.

5.2 Mathematical Reasoning Across Scales and Generations

Consistent gains across models and sampling budgets. Table 1 reports Pass@1 and Pass@8 across three models, two objectives, and five benchmarks. REFT consistently improves performance even on GSM8K, where the models start from strong baselines, and the gains grow larger on Math Avg., showing that REFT is especially effective when additional rollout exploration has more room to pay off (e.g., Qwen2.5-7B GRPO Pass@1: 37.58→43.82). The two metrics measure different things, Pass@1 the quality of a single sample and Pass@8 whether any of eight sampled responses is correct, so their simultaneous improvement shows that REFT benefits both individual samples and the rollout group as a whole. The gains persist across three training seeds (Appendix B.4).

Across scale and generation. Table 2 extends the comparison upward to Qwen3.5-9B, a newer-generation model, and to the larger Qwen2.5-14B-Instruct. Both gain over four Math Avg. points, +5.65 and +4.11, and improve every benchmark. At the other end of the scale, even at 0.5B, where reasoning capacity is limited, REFT also raises all three budgets on GSM8K (Appendix B.2). The benefit extends across both model scale and generation.

Table 2: **Scaling to larger and newer models.** Pass@1 on BigMath-trained models with GRPO (G=16).

Method	MATH-500	AIME24	AIME25	AMC23	Avg.
Qwen3.5-9B	77.00	36.67	20.00	75.00	52.17
GRPO	80.20	40.00	30.00	80.00	57.55
GRPO + REFT	**82.80**	**46.67**	**33.33**	**90.00**	**63.20**
Qwen2.5-14B-Instruct	67.80	10.00	6.67	57.50	35.49
GRPO	75.40	13.33	10.00	65.00	40.93
GRPO + REFT	**76.80**	**16.67**	**16.67**	**70.00**	**45.04**

No trade-off against larger-budget discovery. RLVR can improve Pass@1 by narrowing the policy. As probability concentrates on the trajectories that already succeed, what a larger sampling budget could still discover shrinks with it, and Pass@64 falls even as Pass@1 rises (Yue et al., 2025). Table 3 shows that REFT does not exhibit this trade-off overall. On GSM8K, Pass@64 is already near ceiling and differences are necessarily small, yet REFT still adds +0.59 points averaged over the six model-objective pairs. The effect is clearer on Math Avg., where the average gain reaches +3.13 points. REFT takes the gain without narrowing what a larger budget can still find. Full per-benchmark results are in Appendix B.1.

Table 3: **Coverage at a larger sampling budget.** Pass@64 on GSM8K and Math Avg.

Method	GSM8K	Math Avg.
Qwen2.5-3B-Instruct	98.18	61.95
GRPO	98.64	64.34
GRPO + REFT	**99.39**	**67.82**
DAPO	97.65	65.13
DAPO + REFT	**98.18**	**69.18**
Llama3.2-3B-Instruct	97.42	51.61
GRPO	97.95	55.03
GRPO + REFT	**98.56**	**58.99**
DAPO	97.35	56.28
DAPO + REFT	**98.48**	**59.34**
Qwen2.5-7B-Instruct	97.95	66.26
GRPO	98.33	68.95
GRPO + REFT	**98.94**	**71.54**
DAPO	**98.48**	69.74
DAPO + REFT	98.41	**71.39**

5.3 Comparison with Alternative Diversification Strategies

We compare REFT with six alternatives under a fully matched setup (Qwen2.5-3B-Instruct, GSM8K, GRPO, G=8, LoRA (Hu et al., 2022)), varying only rollout construction. Rollout-MT and Token-MT implement the multi-temperature strategies of Zhuang et al. (2025), respectively cycling temperatures $\{1.0, \ldots, 1.4\}$ across the rollouts of each group and applying T=1.2 to the top 40% high-entropy tokens within a 100K-token rolling window. LATR (Xing et al., 2026) branches at high-uncertainty positions, and prunes insufficiently divergent branches by edit distance. Three targeted controls then localize the advantage. First-token T=2.0 raises temperature only at the first token without guaranteeing coverage, and the two positional controls move the same REFT sampling scheme to the second token or to the highest-entropy position among the first 64 tokens.

Table 4: **Prior diversification methods and targeted controls.** Qwen2.5-3B-Instruct on GSM8K with GRPO (G=8), varying only rollout construction. Controls raise first-token temperature or apply the same REFT sampling scheme to the second token or the highest-entropy position among the first 64 tokens.

Method	Pass@1	Pass@8	Pass@64
Qwen2.5-3B-Instruct	80.14	94.16	98.18
GRPO	84.53	95.38	98.64
GRPO + REFT	**86.28**	**96.13**	**99.39**
Rollout-MT	85.06	95.45	98.33
Token-MT	85.67	95.68	97.95
LATR	84.61	96.06	97.73
First-token T=2.0	85.59	95.83	98.56
Second-token control	85.29	95.60	98.48
Max-entropy control	85.37	95.45	97.95

Two patterns stand out in Table 4. (i) Only REFT improves at every budget. Each alternative improves Pass@1 and Pass@8 at the matched budget but lands below the GRPO baseline at Pass@64, reproducing the trade described by Yue et al. (2025). The best alternative changes with the budget, Token-MT at Pass@1, LATR at Pass@8, and First-token T=2.0 at Pass@64, whereas REFT leads at all three. All these come at the lowest rollout time per step, roughly $1.00\times$ GRPO in our runs versus 1.34–$4.81\times$ for the prior methods (Appendix C.4). (ii) The controls localize the gain to explicit coverage at the first token. First-token T=2.0 falls short despite intervening at the same position, since temperature does not ensure coverage of low-probability alternatives in a small group. Applying the same REFT sampling scheme at the second token confirms that early routing itself is beneficial, but it remains clearly below REFT, as the second token is already conditioned on the sampled first token. Moving further, the max-entropy control performs comparably to or below even the second-token control, as the intervention now lands inside an ongoing reasoning step. The effect grows as the intervention approaches the very first choice, a position prior methods have largely overlooked. Descriptions and implementation details of the three prior methods are in Appendix C.2.

5.4 Code Generation

Table 5 extends REFT to code generation, where it improves on HumanEval+, MBPP+, and LCB v4 for both models, yielding gains in all twelve reported comparisons. The two models expose different kinds of first semantic choices. DeepSeek-R1-Distill-Qwen-7B produces an explicit reasoning trace, so REFT branches on the first semantic token after `<think>`, following the model's official usage recommendation. As in mathematical reasoning, this token is typically a discourse opener. Qwen2.5-Coder-3B-Instruct instead generates the program directly, so REFT branches on its first semantic code token. Unlike the former, the first semantic code token can already influence the program's outer organization, such as whether generation begins with an import, a function definition, or an input-handling construct. At the same time, it is chosen before most program-specific commitments have been made, so it remains low-load relative to the tokens that follow. As generation proceeds, syntax, variable bindings, control flow, and earlier implementation decisions increasingly constrain what can follow. REFT introduces diversity before these dependencies accumulate, varying how a program is organized while leaving the policy to develop each continuation coherently on its own. Rollout examples are shown in Appendix H.

Table 5: **Performance on code-generation RLVR.** GRPO versus GRPO + REFT (G=8) with sandbox-test rewards. REFT improves all twelve comparisons.

	HumanEval+		MBPP+		LCB v4	
Method	Pass@1	Pass@8	Pass@1	Pass@8	Pass@1	Pass@8
DeepSeek-R1-Distill-Qwen-7B	70.5	84.8	60.8	76.2	21.0	30.9
GRPO	72.0	86.6	61.9	78.3	22.3	32.5
GRPO + REFT	**73.2**	**87.8**	**65.1**	**79.9**	**23.1**	**33.7**
Qwen2.5-Coder-3B-Instruct	78.4	89.6	62.5	78.3	10.0	17.5
GRPO	79.3	90.2	63.8	79.1	10.2	17.8
GRPO + REFT	**82.3**	**92.1**	**64.8**	**80.7**	**10.8**	**18.8**

6 Analysis: Why First-Token Coverage Matters

We investigate whether REFT's gains reflect productive exploration rather than changes to first-token concentration alone. We first examine a first-token loss-masking control, then track how first-token concentration evolves during training. Finally, we ask whether the additional exploration more often reaches correct completions rather than merely producing more varied failures.

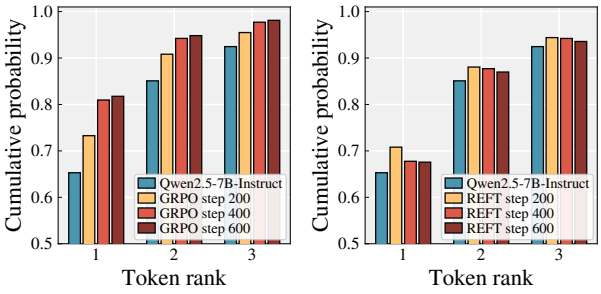
REFT remains effective under first-token loss masking. One possible explanation for REFT's gains is that it reduces direct reinforcement of dominant first tokens. To test whether removing these updates alone is sufficient, we retain standard GRPO sampling but set the policy-loss weight of the first token to zero. The sampling rule and the loss for subsequent tokens are unchanged. As shown in Table 6, this control remains close to GRPO at Pass@1, Pass@8, and Pass@64. Thus, suppressing direct first-token updates alone

Table 6: **First-token loss masking.** The mask sets the policy-loss weight of the first token to zero. GRPO with the mask stays close to GRPO, whereas REFT with the same mask improves over masked GRPO at every budget (Qwen2.5-3B-Instruct on GSM8K with GRPO).

Method	Pass@1	Pass@8	Pass@64
GRPO	84.53	95.38	98.64
GRPO + first-token mask	84.61	95.75	98.64
GRPO + REFT + first-token mask	85.75	95.98	99.01
GRPO + REFT	**86.28**	**96.13**	**99.39**

is insufficient to obtain the gains of REFT. By contrast, applying the same mask under REFT sampling still improves over the masked GRPO control at every budget. Although updates on later tokens may still influence the first-token prior indirectly, these results imply that REFT's gains require learning from the first-token-conditioned trajectories introduced by its broader sampling.

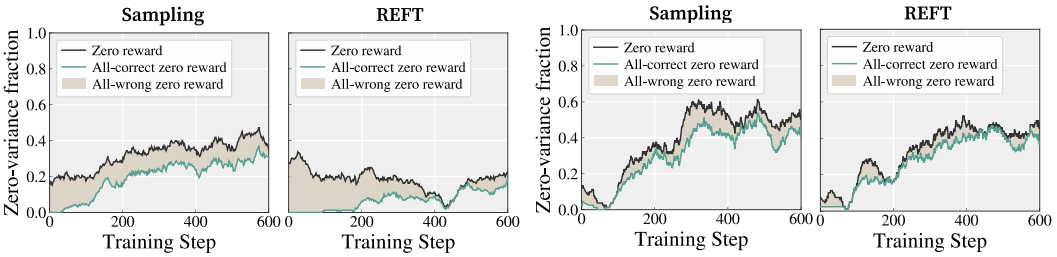
Better performance with less first-token sharpening. We next examine the learned policy's first-token distribution. Under standard GRPO, the top-1 first-token probability increases during training (Figure 5a), whereas it remains comparatively stable under REFT (Figure 5b). Thus, REFT achieves better performance without further sharpening its preference for dominant first tokens. Together with the masking control, this supports the interpretation that REFT benefits from learning on underexplored continuations, rather than merely reinforcing already preferred starting tokens.



(a) Standard GRPO (b) GRPO + REFT

Figure 5: **Less first-token sharpening under REFT.** Top-r cumulative first-token probability during training (Qwen2.5-7B-Instruct with GSM8K). REFT keeps the prior flatter than GRPO.

Broader coverage reaches successful completions. Greater diversity need not improve exploration. A rollout group may contain many different but uniformly unsuccessful trajectories. We therefore examine how often a group fails to produce any correct completion. With binary rewards, zero-variance groups are either all-wrong or all-correct. Although neither provides within-group reward contrast, only an all-wrong group exhausts its rollout budget without a single success. Figure 6 shows that REFT's all-wrong rate decreases as training proceeds and falls below that of standard sampling, even when the overall zero-variance rates are comparable. Thus, REFT more often finds at least one correct completion within the same rollout budget, what Pass@k measures, suggesting that its added diversity supports productive exploration rather than merely increasing variation among failure cases.



(a) Qwen2.5-0.5B-Instruct, GSM8K, DAPO (b) Qwen2.5-3B-Instruct, BigMath-Easy, GRPO

Figure 6: **REFT reduces all-wrong groups.** Each panel shows the fraction of zero-variance groups (black) and all-correct subset (green) for standard sampling (left) and REFT (right). The shaded gap is the all-wrong subset.

7 Conclusion

We study how to diversify a fixed rollout budget in RLVR and identify the first token of the response as a surprisingly efficient exploration site. Our diagnosis reveals a sharply concentrated first-token prior even though lower-probability alternatives yield distinct continuations with comparable downstream correctness, leaving viable trajectories underexplored. REFT exploits this mismatch by explicitly covering first tokens drawn uniformly from the policy's top candidates, allocating the fixed rollout budget across them while leaving the rest of the RLVR pipeline unchanged. Across eight models from 0.5B to 14B in the Qwen, Llama, and DeepSeek-R1-Distill families, REFT consistently improves finite-budget performance on mathematical reasoning and code generation, on benchmarks including GSM8K, MATH-500, AIME, HumanEval+, and LiveCodeBench. The gains hold under both GRPO and DAPO and across Pass@1, Pass@8, and Pass@64, and REFT also outperforms alternative diversification strategies under a matched rollout budget. Overall, useful exploration in RLVR can arise not only at uncertain reasoning pivots, but also from choices the model already treats as settled.

AI USE STATEMENT

We used generative AI tool only to assist with writing, such as correcting grammar and polishing wording. All AI-assisted text was reviewed and edited by the authors, who take full responsibility for the final content of this work.

ETHICS STATEMENT

This work studies rollout construction for reinforcement learning with verifiable rewards on mathematical reasoning and code generation. It does not involve human subjects, personal data, or crowdsourced annotation. All models and datasets are publicly available and used under their respective licenses (Tables 14 and 17). Generated programs are executed only in a sandboxed environment against provided unit tests.

REPRODUCIBILITY STATEMENT

Section 4 and Algorithm 1 specify REFT in full, and Appendix C.4 describes how it is implemented in a standard vLLM pipeline without engine modification. Section 5.1 summarizes the experimental setup, and Appendix C provides the full training and evaluation configurations, including hyperparameters, model and dataset sources, and hardware. Appendix D defines the semantic-diversity metric, Appendix E details the training objective, and Appendix B.4 reports three-seed repetitions of the core comparisons. Appendix C.2 specifies the configurations of the compared prior methods, and Appendix B.5 gives the protocol for the general-utility evaluation. We will release our code upon publication.

REFERENCES

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, et al. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*, 2025.

Bizhe Bai, Xinyue Wang, Peng Ye, and Tao Chen. Learning to explore with parameter-space noise: A deep dive into parameter-space noise for reinforcement learning with verifiable rewards. *arXiv preprint arXiv:2602.02555*, 2026.

Chi-Chih Chang, Siqi Zhu, Zhichen Zeng, Haibin Lin, Jiaxuan You, Mohamed S Abdelfattah, Ziheng Jiang, and Xuehai Qian. Srt: Accelerating reinforcement learning via speculative rollout with tree-structured cache. *arXiv preprint arXiv:2601.09083*, 2026.

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

Zhuoming Chen, Hongyi Liu, Yang Zhou, Haizhong Zheng, and Beidi Chen. Jackpot: Optimal budgeted rejection sampling for extreme actor-policy mismatch reinforcement learning. *arXiv preprint arXiv:2602.06107*, 2026.

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

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Xingyu Dang, Christina Baek, J Zico Kolter, and Aditi Raghunathan. Assessing diversity collapse in reasoning. In *ICLR 2025 Workshop on SSI-FM*, 2025.

Yangyi Fang, Jiaye Lin, Xiaoliang Fu, Cong Qin, Haolin Shi, Chaowen Hu, Lu Pan, Ke Zeng, and Xunliang Cai. How to allocate, how to learn? dynamic rollout allocation and advantage modulation for policy optimization. *arXiv preprint arXiv:2602.19208*, 2026.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Hao Gu, Hao Wang, Jiacheng Liu, Lujun Li, Qiyuan Zhu, Bei Liu, Binxing Xu, Lei Wang, Xintong Yang, Sida Lin, et al. Qarl: Rollout-aligned quantization-aware rl for fast and stable training under training–inference mismatch. *arXiv preprint arXiv:2604.07853*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. TreeRL: LLM reinforcement learning with on-policy tree search. In *ACL*, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

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

Zican Hu, Shilin Zhang, Yafu Li, Jianhao Yan, Xuyang Hu, Leyang Cui, Xiaoye Qu, Chunlin Chen, Yu Cheng, and Zhi Wang. Diversity-incentivized exploration for versatile reasoning. In *ICLR*, 2026.

Baizhou Huang and Xiaojun Wan. Pros: Towards compute-efficient RLVR via rollout prefix reuse. In *ICLR*, 2026.

Wei Huang, Yi Ge, Shuai Yang, Yicheng Xiao, Huizi Mao, Yujun Lin, Hanrong Ye, Sifei Liu, Ka Chun Cheung, Hongxu Yin, et al. Qerl: Beyond efficiency–quantization-enhanced reinforcement learning for llms. In *ICLR*, 2026.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *ICLR*, 2025.

Constantinos Karouzos, Xingwei Tan, and Nikolaos Aletras. Where does output diversity collapse in post-training? *arXiv preprint arXiv:2604.16027*, 2026.

Minwu Kim, Safal Shrestha, and Keith Ross. Training reasoning models on saturated problems via failure-prefix conditioning. *arXiv preprint arXiv:2601.20829*, 2026.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of rlhf on LLM generalisation and diversity. In *ICLR*, 2024.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *SOSP*, 2023.

Thanh-Long V Le, Myeongho Jeon, Kim Vu, Viet Lai, and Eunho Yang. No prompt left behind: Exploiting zero-variance prompts in llm reinforcement learning via entropy-guided advantage shaping. In *ICLR*, 2026.

Jia Li, Edward Beeching, Lewis Tunstall, Benjamin Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Leandro Yu, Albert Q. Jiang, Zhiqiang Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. https://huggingface.co/datasets/AI-MO/NuminaMath-TIR, 2024. Hugging Face dataset repository.

Ziheng Li, Zexu Sun, Jinman Zhao, Erxue Min, Yongcheng Zeng, Hui Wu, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Xu Chen, et al. Staying in the sweet spot: Responsive reasoning evolution via capability-adaptive hint scaffolding. *arXiv preprint arXiv:2509.06923*, 2025.

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 *ICLR*, 2024.

Bingshuai Liu, Ante Wang, Zijun Min, Liang Yao, Haibo Zhang, Yang Liu, Anxiang Zeng, and Jinsong Su. Spec-rl: Accelerating on-policy reinforcement learning via speculative rollouts. *arXiv preprint arXiv:2509.23232*, 2025a.

Chenxi Liu, Junjie Liang, Yuqi Jia, Bochuan Cao, Yang Bai, Heng Huang, and Xun Chen. Explore data left behind in reinforcement learning for reasoning language models. *arXiv preprint arXiv:2511.04800*, 2025b.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *NeurIPS*, 2023.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *NeurIPS*, 2025c.

Open R1. Verifiable coding problems: Python, decontaminated, tested, and shuffled. https://huggingface.co/datasets/open-r1/verifiable-coding-problems-python_decontaminated, 2025. Hugging Face dataset repository.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL https://qwen.ai/blog?id=qwen3.6-27b.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: memory optimizations toward training trillion parameter models. In *SC20*, 2020.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 2019.

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

Amrith Setlur, Zijian Wang, Andrew Cohen, Paria Rashidinejad, and Sang Michael Xie. Reuse your flops: Scaling rl on hard problems by conditioning on very off-policy prefixes. *arXiv preprint arXiv:2601.18795*, 2026.

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

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In *NeurIPS*, 2020.

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, 2023.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Zhongwei Wan, Yun Shen, Zhihao Dou, Donghao Zhou, Yu Zhang, Xin Wang, Hui Shen, Jing Xiong, Chaofan Tao, Zixuan Zhong, et al. Dsdr: Dual-scale diversity regularization for exploration in llm reasoning. *arXiv preprint arXiv:2602.19895*, 2026.

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

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS*, 2024.

Longxuan Wei, Yubo Zhang, Zijiao Zhang, Zhihu Wang, Shiwan Zhao, Tianyu Huang, Huiting Zhao, Chenfei Liu, Shenao Zhang, and Junchi Yan. Entropy-tree: Tree-based decoding with entropy-guided exploration. *arXiv preprint arXiv:2601.15296*, 2026.

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

Shangyu Xing, Siyuan Wang, Chenyuan Yang, Xinyu Dai, and Xiang Ren. Lookahead tree-based rollouts for enhanced trajectory-level exploration in reinforcement learning with verifiable rewards. In *ICLR*, 2026.

Yixuan Even Xu, Yash Savani, Fei Fang, and J Zico Kolter. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Chenghao Yang, Lin Gui, Chenxiao Yang, Victor Veitch, Lizhu Zhang, and Zhuokai Zhao. Let it calm: Exploratory annealed decoding for verifiable reinforcement learning. *arXiv preprint arXiv:2510.05251*, 2025.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Juncai Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *NeurIPS*, 2025.

Song Yu, Li Li, Wenwen Zhao, and Zhisheng Yang. Erpo: Token-level entropy-regulated policy optimization for large reasoning models. *arXiv preprint arXiv:2603.28204*, 2026.

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

Yi Zhang and Math-AI. American invitational mathematics examination (aime) 2024. https://huggingface.co/datasets/math-ai/aime24, 2024. Hugging Face dataset repository.

Yi Zhang and Math-AI. American invitational mathematics examination (aime) 2025. https://huggingface.co/datasets/math-ai/aime25, 2025. Hugging Face dataset repository.

Yuheng Zhang, Wenlin Yao, Changlong Yu, Yao Liu, Qingyu Yin, Bing Yin, Hyokun Yun, and Lihong Li. Improving sampling efficiency in rlvr through adaptive rollout and response reuse. *arXiv preprint arXiv:2509.25808*, 2025.

Ziqi Zhao, Zhaochun Ren, Jiahong Zou, Liu Yang, Zhiwei Xu, Xuri Ge, Zhumin Chen, Xinyu Ma, Daiting Shi, Shuaiqiang Wang, et al. Reinforced efficient reasoning via semantically diverse exploration. *arXiv preprint arXiv:2601.05053*, 2026.

Haizhong Zheng, Yang Zhou, Brian R Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *NeurIPS*, 2025a.

Tianyu Zheng, Tianshun Xing, Qingshui Gu, Taoran Liang, Xingwei Qu, Xin Zhou, Yizhi Li, Zhoufutu Wen, Chenghua Lin, Wenhao Huang, et al. First return, entropy-eliciting explore. *arXiv preprint arXiv:2507.07017*, 2025b.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

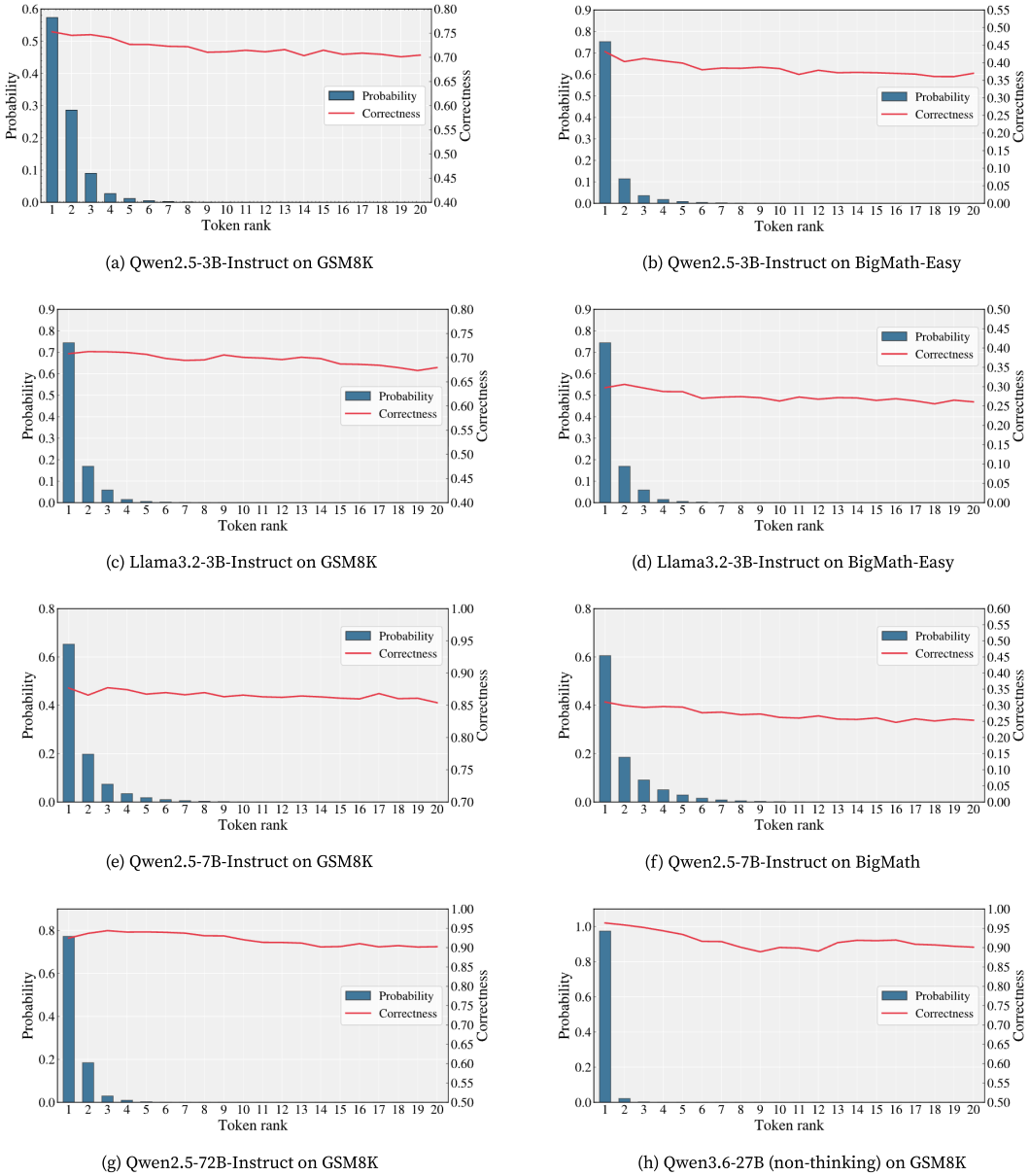
Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in llm reasoning. In *NeurIPS*, 2025.

Haomin Zhuang, Yujun Zhou, Taicheng Guo, Yue Huang, Fangxu Liu, Kai Song, and Xiangliang Zhang. Exploring multi-temperature strategies for token-and rollout-level control in RLVR. *arXiv preprint arXiv:2510.08892*, 2025.

A Extended First-Token Diagnosis

A.1 Probability and Correctness Across Models

A natural objection is that the pattern in Figure 1 may be an artifact of a single model–dataset pair. Figure 7 and Table 7 extend the core diagnosis across datasets, model families, scales, and recent checkpoints. The pattern persists throughout. Probability mass remains heavily concentrated on the highest-ranked first tokens, while forced-first-token correctness stays comparatively flat across the top-20 alternatives. This supports the diagnosis in Section 3 that the first-token concentration is a recurring phenomenon rather than an artifact of the particular setting in Figure 1.



(a) Qwen2.5-3B-Instruct on GSM8K

(b) Qwen2.5-3B-Instruct on BigMath-Easy

(c) Llama3.2-3B-Instruct on GSM8K

(d) Llama3.2-3B-Instruct on BigMath-Easy

(e) Qwen2.5-7B-Instruct on GSM8K

(f) Qwen2.5-7B-Instruct on BigMath

(g) Qwen2.5-72B-Instruct on GSM8K

(h) Qwen3.6-27B (non-thinking) on GSM8K

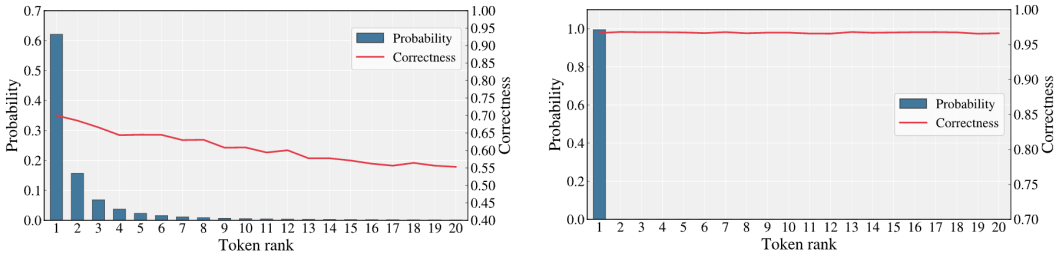
Figure 7: **Probability–correctness decoupling across models and datasets.** Bars show mean first-token probability, and lines show forced-first-token rollout correctness. Across models and scales, probability drops sharply with token rank, whereas correctness remains comparatively flat.

Across the nine model–dataset pairs, the top-1/top-20 probability ratio ranges from about 10^3 to 10^7, while the corresponding correctness gap remains at most 6.3 pp. Additionally, the concentration is particularly extreme in several larger and more recent checkpoints (bottom block of Table 7). Qwen2.5-72B-Instruct exhibits a 1.4×10^7 top-1/top-20 probability ratio with only a 2.2 pp correctness gap. Gemma4-31B-it and Qwen3.6-27B exhibit near-deterministic first-token priors, assigning 99.5% and 97.4% of the probability mass to the top-ranked opener, with top-1/top-20 ratios about 3×10^6 and 2.6×10^5, respectively. Even the largest gap in the table, the 6.3 pp of Qwen3.6-27B, accompanies a five-order-of-magnitude difference in sampling probability. Extremely sharp first-token priors thus arise in larger and more recent checkpoints without a commensurate separation in correctness.

Table 7: **Probability–correctness decoupling across models, datasets, and scales.** p_1 and p_{20} are mean probabilities of the rank-1 and rank-20 first tokens, and $C_1 \to C_{20}$ is forced-first-token rollout correctness at those two ranks. The top-1/top-20 probability ratio ranges from about 10^3 to 10^7, while correctness moves by at most a few points. The bottom block reports larger and more recent models.

Model	Dataset	p_1	p_{20}	p_1/p_{20}	$C_1 \to C_{20}$ (%)	ΔC (pp)
Qwen2.5-3B-Instruct	GSM8K	0.570	2.68×10^{-5}	$\approx 2.1 \times 10^4$	$75.29 \to 70.40$	-4.9
Llama3.2-3B-Instruct	GSM8K	0.739	1.24×10^{-4}	$\approx 6.0 \times 10^3$	$70.83 \to 67.95$	-2.9
Qwen2.5-7B-Instruct	GSM8K	0.653	1.29×10^{-5}	$\approx 5.1 \times 10^4$	$87.69 \to 85.41$	-2.3
Qwen2.5-3B-Instruct	BigMath-Easy	0.711	6.38×10^{-4}	$\approx 1.1 \times 10^3$	$43.11 \to 36.99$	-6.1
Llama3.2-3B-Instruct	BigMath-Easy	0.753	2.88×10^{-5}	$\approx 2.6 \times 10^4$	$28.03 \to 24.45$	-3.6
Qwen2.5-7B-Instruct	BigMath	0.606	4.62×10^{-5}	$\approx 1.3 \times 10^4$	$31.04 \to 25.38$	-5.7
Qwen2.5-72B-Instruct	GSM8K	0.773	5.49×10^{-8}	$\approx 1.4 \times 10^7$	$92.48 \to 90.25$	-2.2
Qwen3.6-27B	GSM8K	0.974	3.77×10^{-6}	$\approx 2.6 \times 10^5$	$96.39 \to 90.13$	-6.3
Gemma4-31B-it	GSM8K	0.995	3.22×10^{-7}	$\approx 3.1 \times 10^6$	$96.66 \to 96.63$	-0.03

The Gemma4-31B base–instruct comparison provides a particularly clear within-family contrast (Figure 8). For the base checkpoint, the first-token distribution is relatively soft, and forced-first-token correctness declines substantially with rank, from 70.01% at rank 1 to 55.30% at rank 20. By contrast, after instruction tuning, the distribution becomes extremely concentrated. The instruction-tuned Gemma4-31B-it assigns 0.995 probability to its rank-1 opener, while the rank-20 probability falls to 3.22×10^{-7}. Despite this dramatic sharpening, correctness is essentially unchanged across the same alternatives, moving from 96.66% at rank 1 to 96.63% at rank 20. Thus, the much sharper prior of the instruction-tuned checkpoint is not accompanied by a corresponding separation in outcome quality. Although this comparison does not isolate any particular post-training stage, it shows that a post-trained checkpoint can exhibit dramatically stronger first-token preference than its base model at the same time as the correctness evidence for that preference disappears.



(a) Gemma4-31B on GSM8K

(b) Gemma4-31B-it on GSM8K

Figure 8: **Gemma4-31B base vs. instruction-tuned on GSM8K.** (a) The Gemma4-31B base checkpoint has a relatively soft first-token distribution, with correctness decreasing across ranks. (b) Gemma4-31B-it concentrates nearly all probability on its top-ranked first token, while correctness remains nearly flat across alternative first tokens. Bars show first-token probability, and lines show forced-first-token correctness.

A.2 Per-Prompt First-Token Distributions

The aggregate first-token distribution in Section 3 averages over many prompts. This appendix shows that the same concentration phenomenon appears at the individual-prompt level. For each prompt, we plot the policy's probability over the top-20 first-token candidates. The horizontal axis lists the actual tokens in decreasing-probability order rather than abstract ranks, making the identity of the dominant first tokens visible. On three GSM8K prompts under Llama3.2-3B-Instruct, almost all probability mass falls on the top few first tokens, with the remaining top-20 tokens carrying negligible probability.

GSM8K prompt 0 **(gold answer:** 18)

Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

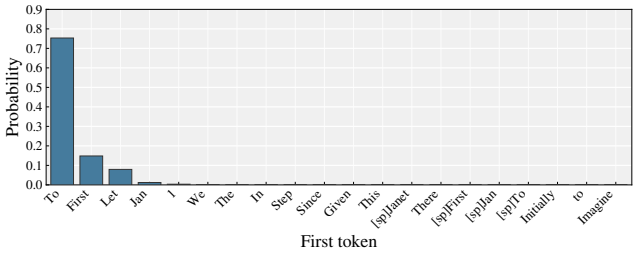


Figure 9: **Top-20 first-token probabilities for GSM8K prompt 0**, under Llama3.2-3B-Instruct.

GSM8K prompt 1 **(gold answer:** 3)

A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?

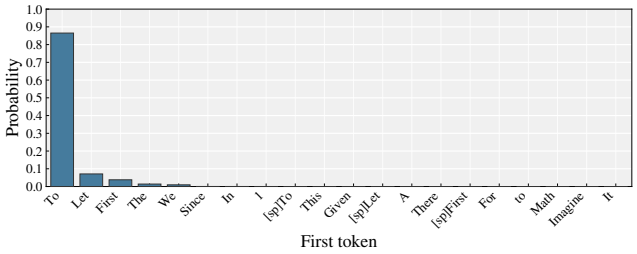


Figure 10: **Top-20 first-token probabilities for GSM8K prompt 1**, under Llama3.2-3B-Instruct.

GSM8K prompt 2 **(gold answer:** 70,000)

Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?

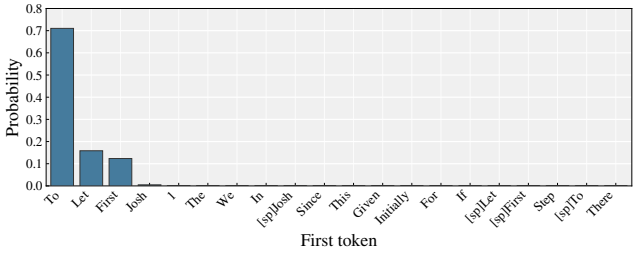


Figure 11: **Top-20 first-token probabilities for GSM8K prompt 2**, under Llama3.2-3B-Instruct.

B Additional Results and Ablations

B.1 Larger-Budget Coverage

Table 8 reports the full per-benchmark Pass@64 results behind the compact summary in Table 3. Pass@64 measures larger-budget coverage, namely whether at least one of 64 sampled completions is correct. Math Avg. averages MATH-500, AIME24, AIME25, and AMC23, excluding GSM8K. Table 8 shows that REFT generally preserves or improves the ability to recover a correct trajectory under a larger sampling budget, rather than trading larger-budget coverage for single-sample accuracy.

Table 8: Pass@64 reasoning performance across models and RLVR methods. Pass@64 denotes whether at least one of 64 sampled completions is correct. Math Avg. excludes GSM8K, and **bold** marks the higher score. Each block compares GRPO and DAPO with their REFT variants.

Method	GSM8K Pass@64	MATH-500 Pass@64	AIME24 Pass@64	AIME25 Pass@64	AMC23 Pass@64	Math Avg. Pass@64
Qwen2.5-3B-Instruct	98.18	87.80	33.33	36.67	90.00	61.95
GRPO	98.64	88.20	36.67	40.00	92.50	64.34
GRPO + REFT	**99.39**	**89.60**	**40.00**	**46.67**	**95.00**	**67.82**
DAPO	97.65	88.00	36.67	43.33	92.50	65.13
DAPO + REFT	**98.18**	**89.20**	**43.33**	**46.67**	**97.50**	**69.18**
Llama3.2-3B-Instruct	97.42	80.60	30.00	13.33	82.50	51.61
GRPO	97.95	81.80	33.33	20.00	85.00	55.03
GRPO + REFT	**98.56**	81.80	**36.67**	**30.00**	**87.50**	**58.99**
DAPO	97.35	82.60	36.67	23.33	82.50	56.28
DAPO + REFT	**98.48**	**83.20**	36.67	**30.00**	**87.50**	**59.34**
Qwen2.5-7B-Instruct	97.95	89.20	40.00	43.33	92.50	66.26
GRPO	98.33	90.80	43.33	46.67	95.00	68.95
GRPO + REFT	**98.94**	**92.00**	**50.00**	46.67	**97.50**	**71.54**
DAPO	**98.48**	90.60	46.67	46.67	95.00	69.74
DAPO + REFT	98.41	**91.40**	46.67	**50.00**	**97.50**	**71.39**

B.2 Results at the Smallest Trained Scale

Table 9 reports GSM8K results for Qwen2.5-0.5B-Instruct, the smallest model we train. Even at 0.5B, where reasoning capacity is limited, REFT improves both GRPO and DAPO at Pass@1, Pass@8, and Pass@64. The direction of the main results thus holds unchanged at the bottom of the scale range.

Table 9: GSM8K performance of Qwen2.5-0.5B-Instruct.

Method	GSM8K Pass@1	GSM8K Pass@8	GSM8K Pass@64
Qwen2.5-0.5B-Instruct	43.37	77.94	90.67
GRPO	51.71	78.17	91.05
GRPO + REFT	**54.21**	**79.68**	**93.71**
DAPO	52.14	78.92	92.80
DAPO + REFT	**53.62**	**79.45**	**94.39**

B.3 Candidate Width and Within-Token Replication

We further verify that REFT is robust to variations in the candidate width Top-N and the number of selected first tokens K. Tables 10 and 11 report ablations on GSM8K with Qwen2.5-3B-Instruct, where the GRPO baseline achieves 84.53 Pass@1 and 95.38 Pass@8. Most REFT variants improve over the baseline, confirming robustness to these hyperparameter choices. Expanding the pool to Top-50 or Top-100 remains competitive in Pass@1 but gradually reduces Pass@8, suggesting that an overly wide candidate pool can spend the finite rollout budget on too many exceedingly rare first tokens. Fixing Top-$N = 20$, increasing K from 1 to 4 raises Pass@1 and Pass@8, showing the benefit of covering multiple first token regions within the same group. Increasing K further to 8 reduces the gain on Pass@8, although Pass@1 remains above the baseline. With $G = 8$, this allocates only one continuation per selected first token, so a viable first token can be misjudged from a single unlucky trace. Setting $K < G$ is therefore not necessary, but it deepens the sample behind each selected token. Our default $K = 4$ assigns two continuations per first token, balancing broader first-token coverage with enough replication to stabilize the rollout signal.

Table 10: Ablation on Top-N with $K = 4$.

N	GSM8K Pass@1	GSM8K Pass@8
10	86.20	95.83
20	**86.28**	**96.13**
50	85.82	95.30
100	85.14	94.92

Table 11: Ablation on K with Top-$N = 20$.

K	GSM8K Pass@1	GSM8K Pass@8
1	84.23	95.15
2	85.51	95.68
4	**86.28**	**96.13**
8	85.82	95.38

B.4 THREE-SEED REPETITIONS

REFT improves the mean in every repeated comparison across the three models, raising Math Avg. by 4.24, 3.10, and 5.07 points. The ranges are well separated on GSM8K, AMC23, and Math Avg. On the 30-question AIME sets, where a single problem shifts the score by 3.33 points, the ranges partly overlap on the 3B models, but the mean direction remains positive. On Qwen2.5-7B-Instruct both sets separate, with means rising by 6.67 and 4.45 points. Thus, the gains are consistent across seeds, not driven by one favorable run.

Table 12: Three-seed repetitions of the core comparisons. Pass@1 as mean±SD over three independent training seeds, under the GRPO objective.

Model	Method	GSM8K	MATH-500	AIME24	AIME25	AMC23	Math Avg.
Qwen2.5-3B-Instruct	GRPO	$84.56_{\pm 0.27}$	$59.73_{\pm 0.50}$	$8.89_{\pm 1.92}$	$4.44_{\pm 1.92}$	$43.33_{\pm 1.44}$	$29.10_{\pm 0.25}$
	GRPO + REFT	$86.23_{\pm 0.31}$	$61.13_{\pm 0.42}$	$15.56_{\pm 1.92}$	$6.67_{\pm 3.33}$	$50.00_{\pm 2.50}$	$33.34_{\pm 1.48}$
Llama3.2-3B-Instruct	GRPO	$75.74_{\pm 0.35}$	$35.53_{\pm 0.50}$	$5.56_{\pm 1.92}$	$2.22_{\pm 1.92}$	$21.67_{\pm 1.44}$	$16.24_{\pm 0.81}$
	GRPO + REFT	$77.69_{\pm 0.31}$	$36.80_{\pm 0.40}$	$6.67_{\pm 3.33}$	$5.56_{\pm 1.92}$	$28.33_{\pm 2.89}$	$19.34_{\pm 1.52}$
Qwen2.5-7B-Instruct	GRPO	$90.95_{\pm 0.32}$	$72.27_{\pm 0.83}$	$14.44_{\pm 1.92}$	$12.22_{\pm 1.92}$	$54.17_{\pm 3.82}$	$38.27_{\pm 1.67}$
	GRPO + REFT	$91.84_{\pm 0.31}$	$73.07_{\pm 0.50}$	$21.11_{\pm 1.92}$	$16.67_{\pm 0.00}$	$62.50_{\pm 2.50}$	$43.34_{\pm 0.80}$

B.5 GENERAL UTILITY RETENTION

Since REFT changes how training rollouts begin, we additionally check whether the trained policy retains its general capabilities outside the RLVR training domain, such as instruction following and general knowledge. We evaluate the final checkpoints on three general-purpose benchmarks using standard inference, with no REFT intervention at evaluation time.

IFEval (Zhou et al., 2023) measures instruction following with prompt-level strict accuracy under zero-shot greedy decoding, and being generation-based, it is the most sensitive of the three to how a response begins. BBH (Suzgun et al., 2023) uses 3-shot normalized accuracy over 24 subtasks, and MMLU-Pro (Wang et al., 2024) uses 5-shot multiple-choice accuracy. Table 13 compares the non-trained baseline model, GRPO, and REFT for Qwen2.5-3B-Instruct trained on GSM8K and BigMath-Easy and for Qwen2.5-14B-Instruct trained on BigMath.

Table 13: **General utility after RLVR training.** All checkpoints are evaluated with standard inference and no REFT intervention. Gray rows report the original model before RLVR training, and parentheses give the RLVR training dataset. Bold marks the higher score between GRPO and GRPO+REFT within each training setting.

Method	IFEval (strict)	BBH (3-shot)	MMLU-Pro (5-shot)
Qwen2.5-3B-Instruct	59.89	46.61	32.71
GRPO (GSM8K)	**60.81**	46.59	**32.92**
GRPO + REFT (GSM8K)	**60.81**	**46.68**	32.75
GRPO (BigMath-Easy)	59.15	46.55	**33.21**
GRPO + REFT (BigMath-Easy)	**60.07**	**46.75**	33.05
Qwen2.5-14B-Instruct	78.56	63.88	49.08
GRPO (BigMath)	78.37	63.70	48.99
GRPO + REFT (BigMath)	**79.85**	**63.74**	**49.09**

Across the three settings, REFT shows no noticeable degradation relative to the original model, scoring within 0.2 points of it or higher on every benchmark. Compared with GRPO, REFT performs comparably on all three benchmarks. Training on math rollouts thus changes general utility little, and REFT improves target-domain performance while retaining it at the level of standard GRPO.

C Experimental Details

C.1 RL Training and Evaluation Details

We evaluate REFT as a drop-in rollout-sampling modification on top of DAPO (Yu et al., 2025) and GRPO (Shao et al., 2024). Table 14 summarizes the models used for RLVR training and diagnosis, together with the sentence encoder used in our semantic-diversity analysis.

Table 14: Summary of models used in this paper. We report the source and license of the base models, together with the sentence encoder used in our semantic-diversity analysis.

Model	Source	License
Qwen2.5-0.5B-Instruct	Qwen (Yang et al., 2024)	Apache License 2.0
Qwen2.5-3B-Instruct	Qwen (Yang et al., 2024)	Qwen Research License
Qwen2.5-7B-Instruct	Qwen (Yang et al., 2024)	Apache License 2.0
Qwen2.5-14B-Instruct	Qwen (Yang et al., 2024)	Apache License 2.0
Qwen2.5-72B-Instruct	Qwen (Yang et al., 2024)	Qwen License
Qwen3.5-9B	Qwen (Team, 2026)	Apache License 2.0
Llama3.2-3B-Instruct	Meta (Grattafiori et al., 2024)	Llama 3.2 Community License
Qwen3.6-27B	Qwen (Qwen Team, 2026)	Apache License 2.0
Gemma4-31B	Google (Team et al., 2026)	Apache License 2.0
Gemma4-31B-it	Google (Team et al., 2026)	Apache License 2.0
DeepSeek-R1-Distill-Qwen-7B	DeepSeek-AI (Guo et al., 2025)	MIT
Qwen2.5-Coder-3B-Instruct	Qwen (Hui et al., 2024)	Qwen Research License
all-mpnet-base-v2	Sentence Transformers (Reimers & Gurevych, 2019; Song et al., 2020)	Apache License 2.0

REFT configuration. Unless otherwise stated, REFT uses candidate width $N = 20$ and selected-token count $K = 4$. For each prompt, four distinct first semantic tokens are sampled uniformly from the policy's top-20 valid candidates, and the fixed rollout budget is allocated evenly across them. Thus, settings with $G = 8$ (GSM8K and code generation) use two continuations per selected first token, and BigMath settings with $G = 16$ use four. The choice $N = 20$ follows the diagnostic result in Section 3, where rollout correctness remains nearly flat across the top-20 first-token ranks even though their probabilities differ by orders of magnitude. The choice $K = 4$ provides coverage of multiple continuation regions while still assigning replicated continuations to each selected first token, reducing sensitivity to a single unlucky rollout. After the selected first token, the remainder of each rollout is generated with the same continuation decoder as the corresponding baseline. Ablations and further discussion are given in Appendix B.3.

Mathematical reasoning. We use GSM8K (Cobbe et al., 2021), BigMath-Easy, and BigMath (Albalak et al., 2025) for mathematical reasoning. GSM8K (Cobbe et al., 2021) contains 7,500 elementary-grade math problems and is used with rollout group size $G = 8$ for all four GSM8K-trained models. For the models with 7 billion parameters or more, we train on the BigMath (Albalak et al., 2025) levels 3–5 subset, containing 121,503 problems. For the 3B-scale models, Qwen2.5-3B-Instruct and Llama3.2-3B-Instruct, we construct BigMath-Easy by restricting BigMath to levels 1–3, yielding 127,945 problems. This follows prior evidence that removing the hardest problems can provide a more effective training signal for smaller models (Albalak et al., 2025). Both BigMath subsets use rollout group size $G = 16$. Maximum completion length is 4,096 tokens for GSM8K and 8,192 tokens for BigMath and BigMath-Easy. Unless explicitly varied by a baseline, all math rollout sampling uses temperature 1.0 and top-$p = 1.0$ (Hu et al., 2025; Huang et al., 2026; Gu et al., 2026; Fang et al., 2026). The verifier reward combines answer accuracy and format, with weights $(2.0, 0.2)$ on GSM8K and $(1.0, 0.2)$ on BigMath and BigMath-Easy. The same reward function and weights are used for the corresponding baseline and REFT runs. Training runs for up to 1,000 steps using LoRA (Hu et al., 2022) with rank 32 and DeepSpeed ZeRO-3 (Rajbhandari et al., 2020) through Hugging Face Accelerate for memory efficiency. Table 15 summarizes the training configuration.

Code generation. We train DeepSeek-R1-Distill-Qwen-7B and Qwen2.5-Coder-3B-Instruct with GRPO ($G = 8$) on `open-r1/verifiable-coding-problems-python_decontaminated-tested-shuffled` (Open R1, 2025), a decontaminated, test-verified

collection of Python programming problems. Generated programs are executed against the provided test cases using the Morph sandbox backend, and the reward is the fraction of tests passed. DeepSeek-R1-Distill-Qwen-7B generates an explicit reasoning trace, whereas Qwen2.5-Coder-3B-Instruct generates Python directly without an explicit reasoning marker. Accordingly, REFT branches on the first semantic token after `<think>` for DeepSeek and on the first semantic code token for Qwen2.5-Coder. Within each model, GRPO and GRPO+REFT use identical optimization and continuation-decoding settings. All code-training rollouts are generated with vLLM (Kwon et al., 2023) in colocate mode. Table 16 summarizes the code-training configuration.

Table 15: Hyperparameters for math RLVR training.

Hyperparameter	**Value**
Optimizer	AdamW-8bit
Training batch size	128
Policy learning rate	$5e{-6}$
LoRA rank	32
Rollouts per prompt	8 for GSM8K, 16 for BigMath / BigMath-Easy
Max response length	4096 for GSM8K, 8192 for BigMath / BigMath-Easy
Rollout temperature	1.0
Rollout top-p	1.0
Clip range $\epsilon_{\text{low}}, \epsilon_{\text{high}}$	0.2, 0.28
Policy updates per rollout	4 for GRPO, 1 for DAPO
Reward weights	GSM8K: Accuracy 2.0, Format 0.2
	BigMath / BigMath-Easy: Accuracy 1.0, Format 0.2
Training step	1,000 steps

Table 16: Hyperparameters for code-generation RLVR training.

Hyperparameter	**Value**
RL objective	GRPO
Optimizer	AdamW-8bit
Policy learning rate	$5e{-6}$
LoRA rank	32
Rollouts per prompt	8
Max prompt length	2048
Max response length	8192
Rollout temperature	Qwen2.5-Coder: 1.0, DeepSeek-R1-Distill: 0.6
Rollout top-p	Qwen2.5-Coder: 1.0, DeepSeek-R1-Distill: 0.95
Clip range $\epsilon_{\text{low}}, \epsilon_{\text{high}}$	0.2, 0.28
Policy updates per rollout	1
Training step	1,000 steps

Evaluation datasets and inference. GSM8K-trained models are evaluated on the full GSM8K test set (Cobbe et al., 2021). BigMath and BigMath-Easy trained models are evaluated on the full evaluation sets of MATH-500 (Lightman et al., 2024), AIME 2024 (Zhang & Math-AI, 2024), AIME 2025 (Zhang & Math-AI, 2025), and AMC 2023 (Li et al., 2024). Math inference uses vLLM (Kwon et al., 2023) with temperature 0.6 and top-$p = 0.95$, and we report Pass@1, Pass@8, and Pass@64. For code generation, we evaluate HumanEval+ and MBPP+ using EvalPlus (Liu et al., 2023) and LiveCodeBench v4 using execution-based correctness (Jain et al., 2025). Code inference uses the same sampling configuration, and we report Pass@1 and Pass@8. Each Pass@k value is computed from k sampled responses per problem and indicates whether at least one response is correct. All results are reported at the final checkpoint of each run.

Hardware. We train Qwen2.5-7B/14B-Instruct, Qwen3.5-9B, and DeepSeek-R1-Distill-Qwen-7B on $4\times$ NVIDIA B200 GPUs. All remaining models are trained on $8\times$ NVIDIA L40S GPUs. For consistency, all runs for the same base model use the same hardware configuration.

Table 17: Summary of datasets used in this paper. We report the source and license metadata for both training and evaluation datasets.

Dataset	Source	License
GSM8K	OpenAI (Cobbe et al., 2021)	MIT
BigMath	SynthLabsAI Big-Math-RL-Verified (Albalak et al., 2025)	Apache License 2.0
MATH-500	OpenAI PRM800K / HuggingFaceH4 (Lightman et al., 2024)	MIT (source repo)
AIME 2024	math-ai (Zhang & Math-AI, 2024)	Apache License 2.0
AIME 2025	math-ai (Zhang & Math-AI, 2025)	Apache License 2.0
AMC 2023	NuminaMath eval suite (Li et al., 2024)	Apache License 2.0
Verifiable Python Problems	open-r1 (Open R1, 2025)	Not specified
HumanEval+	EvalPlus (Liu et al., 2023)	Apache License 2.0
MBPP+	EvalPlus (Liu et al., 2023)	Apache License 2.0
LiveCodeBench v4	LiveCodeBench (Jain et al., 2025)	MIT
IFEval	Google Research (Zhou et al., 2023)	Apache License 2.0
BBH	BIG-Bench Hard (Suzgun et al., 2023)	MIT
MMLU-Pro	TIGER-Lab (Wang et al., 2024)	MIT

C.2 Baseline Implementations for the Head-to-Head Comparison

All alternatives in Section 5.3 share the data, prompts, reward, rollout budget (G=8), GRPO objective, LoRA configuration, and optimization settings of Appendix C.1, and differ only in how training rollouts are constructed. The sampling settings below refer to training rollouts, and all methods are evaluated with the common decoding setting of Appendix C.1.

Rollout-MT. Rollout-level multi-temperature sampling of Zhuang et al. (2025) diversifies a group by sampling its rollouts at different temperatures, so conservative and more exploratory rollouts coexist in the same group rather than all sharing a single temperature. We use the temperature set $\{1.0, 1.1, 1.2, 1.3, 1.4\}$, the in-range mixture that the authors identify as the stable regime. Following the official implementation, temperatures are assigned cyclically to the G=8 rollouts of each group, giving $(1.0, 1.1, 1.2, 1.3, 1.4, 1.0, 1.1, 1.2)$, and each rollout keeps its temperature for every token.

Token-MT. Token-level temperature scheduling of Zhuang et al. (2025) concentrates exploration on high-entropy positions, raising the temperature only where the policy is uncertain. Thus, rollouts are more likely to diverge at these decision points, and low-entropy tokens are sampled as usual. We use the default configuration of the official implementation, with T_{low}=1.0, T_{high}=1.2, a high-entropy ratio of 40%, and an entropy window over the most recent 100,000 generated token positions. At every decoding step, the entropy of the full-vocabulary next-token distribution is computed from the untempered logits and added to the window. Positions at or above the 60th percentile of the window, roughly the top 40%, are sampled at T_{high}, and all others at T_{low}.

LATR. Lookahead tree-based rollouts of Xing et al. (2026) enforce diversity structurally rather than through sampling noise. At uncertain positions, the group branches onto alternative candidate tokens, each new branch is extended for a short lookahead, and branches that remain too similar to their parent are pruned, thus only genuinely divergent trajectories survive in the group. We use hyperparameters reported in the paper which coincide with the defaults of the official implementation. At every generation step, each active branch continues with its top-1 token, and a new branch is opened on the top-2 token when its probability both exceeds the absolute threshold τ_{abs}=0.25 and lies within τ_{rel}=0.15 of the top-1 probability. Each new branch is compared with its parent after 20, 30, and 50 lookahead tokens, and it is pruned together with its descendants if the normalized token edit distance falls below τ_{ed}=0.4 at any of the three windows. We also adopt the hybrid rollout schedule of the paper, which assigns $k_{\text{LATR}}=\lfloor \eta_t G \rceil$ rollouts of each group to the tree and the rest to standard stochastic sampling, with η_t=0.995^t at training step t as reported for the math setting. The tree is therefore capped at k_{LATR} live branches per prompt, and once the cap is reached, lower-probability branches are suppressed. Following the official math training script, tree construction stops early once the branch set is full, using a fullness threshold that starts at 0.98 and decays linearly with rate 0.05 over the generation, and in any case after 1,024 tokens. After building the tree prefix, colocated vLLM completes every sequence at T=1.0 and top-p=1.0. The remaining $G - k_{\text{LATR}}$ rollouts are sampled by the same generator up to the same prefix length before the vLLM continuation.

C.3 Runtime Overhead

Runtime efficiency. REFT does not increase the rollout budget. It only changes how the first token is allocated within the same rollout group. As shown in Table 18, this makes the method essentially cost-free in practice, and the measured GPU-hours are slightly lower than those of standard sampling. In the profiled runs, REFT often reaches higher accuracy (Tables 1 and 9) with shorter average completions, reducing generation length enough to offset the small overhead of first-token routing.

Table 18: Runtime and completion-length changes of REFT. Negative relative differences indicate that REFT is lower than standard sampling. Mean relative differences are computed from the mean values in each column.

	Runtime	Completion Length		
Configuration	GPU-hour Rel. Δ	Sampling Avg.	REFT Avg.	Rel. Δ
GSM8K, Qwen2.5-3B-Instruct, DAPO	-0.02%	245.55	235.87	-3.94%
BigMath-Easy, Qwen2.5-3B-Instruct, GRPO	-0.80%	366.25	342.33	-6.53%
BigMath-Easy, Qwen2.5-3B-Instruct, DAPO	-0.42%	353.18	333.26	-5.64%
Mean	$\mathbf{-0.41\%}$	**321.66**	**303.82**	$\mathbf{-5.55\%}$

C.4 REFT in vLLM-Style Decoding

REFT requires no engine modification. Candidate extraction uses standard log-probability outputs, and forcing a first token amounts to appending one token id to the prompt ids of its assigned rollouts. Measured rollout time per step is $\approx 1.00\times$ standard sampling (Section 5.3), since the only added work is candidate extraction at a single position. For the trained baselines of Section 5.3, rollout time per step in the same matched implementation is $1.34\times$ for Token-MT (token-level entropy tracked over a 100K-token rolling window at every decoding step), $3.80\times$ for Rollout-MT (longer high-temperature trajectories), and $4.81\times$ for LATR (lookahead branches that may later be pruned). The max-entropy control additionally requires per-position entropy computation, which we did not profile. Concretely, for each batch of unique prompts, we (i) obtain the top-N first-token candidates and their log-probabilities at the response-start position via a standard log-probability query, (ii) sample K candidates without replacement on the host, (iii) append each selected token id to the prompt token ids of its assigned rollouts, and (iv) generate all G continuations with the standard vLLM decoder (Kwon et al., 2023). No custom logits processor or separate per-branch decoding engine is required. Prefix caching reuses shared prompt KV blocks, after which each branch maintains its own KV state exactly as in ordinary GRPO rollouts. Continuous batching and standard packing are preserved, and the rollout budget and active-sequence count are unchanged. The only extra work is candidate extraction and CPU-side assignment.

C.5 Response-Start Context $c(x)$ per Model

Table 19 instantiates the definition of Section 3 for every model in the paper, including the `<think>`-style scaffolds of recent families (Guo et al., 2025; Team, 2026; Qwen Team, 2026). To identify the first semantic token, we skip over the chat template's assistant prefix, structural delimiters, and whitespace-only tokens, and these preceding tokens remain part of the conditioning context $c(x)$.

Table 19: Per-model instantiation of the response-start context.

Model	Scaffolding preceding the first token (included in $c(x)$)
Qwen2.5-{0.5B,3B,7B,14B,72B}-Instruct	assistant prefix + `<think>\n`
Llama3.2-3B-Instruct	assistant prefix + `<think>\n`
Qwen3.5-9B / Qwen3.6-27B (non-thinking)	assistant prefix + empty `<think></think>` pair
Gemma4-31B (base / it)	model-turn prefix (base: none)
DeepSeek-R1-Distill-Qwen-7B	assistant prefix + `<think>\n`
Qwen2.5-Coder-3B-Instruct	assistant prefix

D SEMANTIC-DIVERSITY COMPUTATION

To test whether first-token choice affects the continuation rather than only the first word, we measure cosine semantic diversity after removing the first token itself. Each stripped continuation is embedded with `all-mpnet-base-v2` (Song et al., 2020; Reimers & Gurevych, 2019), following prior works (Kirk et al., 2024; Karouzos et al., 2026; Zhao et al., 2026; Bai et al., 2026) that compute semantic diversity using mean pairwise cosine distance between sentence embeddings.

For a rollout group $\mathcal{G}_x = \{y_i\}_{i=1}^{G}$, let $e(\mathrm{strip}(y_i))$ denote the embedding of the continuation. We define within-prompt semantic diversity as the average pairwise cosine distance:

$$D_{\mathrm{sem}}(\mathcal{G}_x) = \frac{2}{G(G-1)} \sum_{1 \le i < j \le G} \Big[1 - \cos\big(e(\mathrm{strip}(y_i)),\, e(\mathrm{strip}(y_j))\big)\Big]. \tag{3}$$

We report the average of $D_{\mathrm{sem}}(\mathcal{G}_x)$ over prompts. Higher values indicate more semantically diverse continuations within the rollout group.

E ROLLOUT DISTRIBUTION AND TRAINING LOSS

In standard rollout sampling, the first token after $c(x)$ is drawn from the behavior policy,

$$f \sim \pi_{\theta_{\mathrm{old}}}(\cdot \mid c(x)).$$

REFT instead selects first tokens through its top-N allocation rule, giving broader coverage to candidates that would rarely be sampled from the original first-token distribution. Conditional on the selected first token, the continuation is generated from the same behavior policy as in the baseline,

$$z \sim \pi_{\theta_{\mathrm{old}}}(\cdot \mid c(x), f).$$

Thus, the distributional change enters through first-token selection, while the conditional continuation decoder remains unchanged.

We apply the same GRPO/DAPO loss to the resulting rollouts without an additional importance correction for the modified first-token sampling distribution. This keeps the optimization procedure fixed while changing only how the rollout group is constructed. Consequently, the update reflects REFT's deliberate reallocation of the finite rollout budget across plausible first tokens, rather than reweighting the collected trajectories back toward the sharply concentrated original prior. PPO-style clipping remains part of the underlying GRPO/DAPO objective and stabilizes updates between the current and behavior policies. For lower-ranked candidates, whose policy probability is orders of magnitude below $1/N$, an explicit importance correction would strongly downweight rollouts starting from those first tokens and pull the update back toward the concentrated prior that REFT is designed to broaden.

This setup is related in spirit to RLVR interventions that modify rollout collection or conditioning while keeping the optimization fixed, such as temperature-scheduled generation (Zhuang et al., 2025; Yang et al., 2025) and prefix-conditioned or prefix-reuse methods (Setlur et al., 2026; Huang & Wan, 2026; Li et al., 2025; Kim et al., 2026).

Additionally, monitoring training throughout on Qwen2.5-3B-Instruct and GSM8K, reward curves are smooth under both methods. The fraction of policy-loss tokens outside the clipping range stays below 0.02% on average for both (means 8.18×10^{-5} for GRPO and 1.61×10^{-4} for REFT; maxima 1.51×10^{-3} and 2.14×10^{-3}), and the mean gradient norm is higher under REFT (0.049 vs. 0.036), with no gradient spikes, reward collapse, or degenerate generation. Apart from the first-token sampler, the answer verifier, reward function, advantage computation, RL objective, model architecture, continuation decoder, and total rollout budget are otherwise unchanged. Across the experiments in Section 5, this modified rollout distribution trains stably and yields the performance gains reported in the main results.

Appendix B.5 further evaluates general-purpose capabilities under standard inference and finds no noticeable degradation relative to GRPO on instruction following, general reasoning, and knowledge benchmarks.

F Additional Analysis

Training-time diversity. Figure 12 tracks the semantic diversity of the rollout groups each method collects during training, measured on continuation embeddings. REFT yields markedly more diverse groups than standard sampling throughout training. Diversity drifts downward for every method as the policy sharpens, as expected under reward-driven optimization, but REFT remains above the alternatives at every step rather than converging to them. The fixed-token controls show that this gain is not a generic effect of first-token forcing. Fixing $f_{20}(x)$ roughly tracks standard sampling and fixing $f_1(x)$ falls below it, so forcing any single token does not by itself widen the group. Because REFT changes only how rollouts are drawn, the added diversity reflects the broader set of alternatives the policy is exposed to under group-relative updates, while the objective itself still rewards verifier success rather than output dispersion. Read alongside Tables 1 and 3, these results indicate that sustained exploration during training accompanies higher single-sample accuracy and larger-budget success rates.

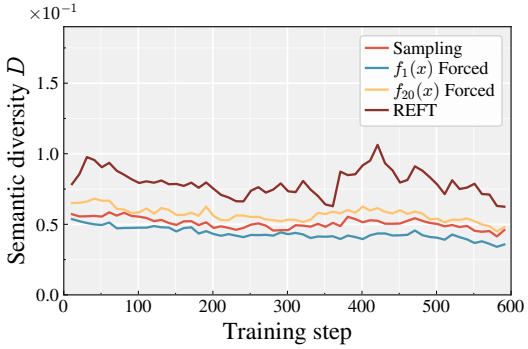


Figure 12: **REFT broadens exploration throughout training.** Semantic diversity D of rollout groups collected during training, measured as the dispersion of continuation embeddings within each group. REFT stays above standard sampling and the fixed first-token controls $f_1(x)$ and $f_{20}(x)$ at every step, and the gap persists as all curves gradually decline.

Outcome diversity beyond trajectory diversity. The near-flat first-token-conditioned accuracy in Section 3 shows viability, not equivalence. Plausible alternative first tokens are not destructive to correctness on average, but they can still route generation into different continuation regions. We therefore complement trajectory-level semantic diversity with final-answer diversity, a tractable proxy for outcome-space coverage used in prior work on reasoning diversity (Song et al., 2025; Dang et al., 2025). Figure 13 shows that REFT produces more distinct answers than standard sampling during training, suggesting that first token diversification changes not only the reasoning text but also the set of solution attempts under the same rollout budget. More distinct answers alone do not establish useful exploration. This diagnostic complements the analysis in Section 6, where the zero-variance decomposition in Figure 6 shows fewer all-wrong zero-variance groups under REFT in reported comparisons.

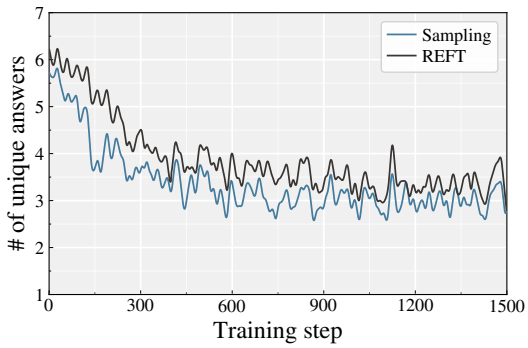


Figure 13: **Within-group answer diversity.** Number of unique final answers per rollout group during training on Qwen2.5-0.5B-Instruct, GSM8K, DAPO.

G Qualitative Math Rollout Examples

We present two logged GSM8K rollout examples illustrating the rollout-level effect of REFT. Both examples are sampled from Qwen2.5-3B-Instruct trained with GRPO on GSM8K. For each prompt, we contrast eight standard-sampling rollouts with eight REFT rollouts produced as $K{=}4$ first-token branches with $G/K{=}2$ continuations per branch. We display representative continuations from each condition rather than all sixteen completions.

G.1 Case 1: Water Consumption Problem (gold answer = 26)

Prompt

John likes to have a glass of water with breakfast, lunch and dinner. Finally, he has one before he goes to bed as well. John does this every weekday, but on the weekends he likes to relax and have a soda with dinner instead. How many glasses of water does John drink in a week?

Standard sampling: $0/8$ correct. The rollouts open almost exclusively with `To` or `First`, and all eight reach incorrect answers ($7, 15, 17, 17, 20, 22, 22, 22$). They make several distinct bookkeeping errors. Some, like rollout 1 below, undercount the daily water occasions, while others, like rollout 6, identify the four weekday occasions correctly but undercount weekend consumption after replacing dinner with soda.

Standard rollout 1

<think>
To determine the total number of glasses of water John drinks in a week, we need to break down his water consumption pattern and calculate accordingly.
John drinks one glass of water with each meal throughout the week: - For breakfast - For lunch - For dinner
[. . .]
On weekends (Saturday and Sunday), he drinks a soda instead of water during dinner: - Saturday: 1 glass of water - Sunday: 1 glass of water
5 (weekdays) + 2 (weekend days) = 7 glasses
</think> <answer> 7 </answer>

Standard rollout 6

<think>
To determine how many glasses of water John drinks in a week, we need to break down his drinking habits by weekdays and weekends.
1. **Weekdays**: John drinks water with breakfast, lunch, dinner, and before bed. That gives 4 glasses per weekday $\times 5$ weekdays $= 20$ glasses.
[. . .]
2. **Weekends**: He has a soda with dinner on weekend days, but still drinks water at the other meals.
$20 + 2 = 22$ glasses
</think> <answer> 22 </answer>

REFT: $8/8$ correct. REFT allocates its eight rollouts across four selected first-token branches, `Let`, `First`, `Given`, and `Initially`, with two continuations each. Each opens a different framing of the same problem (step-by-step decomposition, weekday-first, premise-restatement, sequential narration), and all eight continuations reach the correct answer 26.

REFT rollout 1 First Token = `Let`

<think>
Let's break down the problem step by step.
1. John drinks water at the following times: breakfast, lunch, dinner, and before bed. 2. Therefore, John drinks water 4 times a day. 3. There are 5 weekdays and 2 weekend days. 4. On weekdays: $5 \times 4 = 20$ glasses. 5. On weekends, he switches from water to soda for dinner, so he drinks 3 glasses each weekend day, $2 \times 3 = 6$ glasses.
$20 + 6 = 26$ glasses.
</think> <answer> 26 </answer>

REFT rollout 5 **First Token = `Given`**

<think>
Given the information, John drinks water with his meals every weekday and a soda with his dinner on weekends. A week consists of 5 weekdays and 2 weekend days.
[. . .]
Weekdays contribute $5 \times 4 = 20$ glasses, and weekend days contribute $2 \times 3 = 6$ glasses, for a total of 26 glasses.
</think> <answer> 26 </answer>

REFT rollout 7 **First Token = `Initially`**

<think>
Initially, let's determine how many glasses of water John drinks on a regular weekday. According to the problem, John drinks one glass at breakfast, lunch, dinner, and before bed, giving 4 glasses per weekday.
[. . .]
For weekends, dinner is replaced by soda, leaving 3 glasses of water per day. Total: $5 \times 4 + 2 \times 3 = 26$ glasses.
</think> <answer> 26 </answer>

Why this case is informative. Standard sampling does not fail by collapsing onto a single wrong answer. Its rollouts make several distinct bookkeeping errors, and none keeps the weekday and weekend accounting consistent at the same time. REFT's alternative first-token conditions more often route into decompositions that keep this accounting consistent, and here all eight continuations reach the correct total, with each displayed rollout separating the four weekday occasions from the three weekend ones. Since everything after the opener is sampled with the same decoder under both conditions, this outcome is consistent with the routing effect described in Section 3.

G.2 Case 2: Lemon Tree Problem (gold answer = 13)

Prompt

Carlos is planting a lemon tree. The tree will cost $90 to plant. Each year it will grow 7 lemons, which he can sell for $1.5 each. It costs $3 a year to water and feed the tree. How many years will it take before he starts earning money on the lemon tree?

Standard sampling: $0/8$ correct. Five of the eight rollouts return the same wrong answer of 12 (computing the year in which cumulative profit is non-negative, rather than the year in which it is strictly positive); the remaining three return 9, 73.5, and 1. The five rollouts that answer 12 differ in surface text but share the same off-by-one error.

Standard rollout 1

<think>
To determine how many years it will take before Carlos starts earning money on his lemon tree, we need to set up an inequality based on his profit.
[. . .]
$7.5y \geq 90 \Rightarrow y \geq 12$
So Carlos will start earning money in year 12.
</think> <answer> 12 </answer>

REFT: $7/8$ correct. REFT allocates its eight rollouts across four selected first-token branches, `First`, `Carlos`, `To`, and `When`, with two continuations each. Seven of the eight rollouts correctly compute that the tree starts *strictly earning* money in year 13.

REFT rollout 1 **First Token = `First`**

<think>
First, we need to determine if the lemon tree will generate more money through sales than it costs to maintain.
[. . .]
Annual net profit: $7 \times 1.5 - 3 = 7.5$. We need $7.5y > 90$, i.e., $y > 12$. The smallest such integer is $y = 13$.
</think> <answer> 13 </answer>

REFT rollout 3 First Token = `Carlos`

<think>
Carlos starts by investing \$90 to plant the lemon tree. Each year, he has to spend \$3 to water and feed it, while earning $7 \times \$1.5 = \$10.5$ from selling lemons.
[. . .]
Cumulative net profit after y years is $7.5y - 90$, which becomes positive at $y = 13$.
</think> <answer> 13 </answer>

REFT rollout 7 First Token = `When`

<think>
When calculating the number of years it will take before Carlos starts earning money on the lemon tree, we need to consider both the initial cost and the recurring annual costs and income.
[. . .]
The break-even is at $y = 12$, so Carlos first *earns* money in year 13.
</think> <answer> 13 </answer>

Takeaway. Across both cases, the difference between standard sampling and REFT is not that REFT produces stranger or longer text. The completions remain coherent English with valid <think>/<answer> structure and no degenerate behavior. What changes is that diversifying the first token spreads the group over several framings of the same problem, and these framings can differ in how they set up the computation and where they conclude. These examples illustrate at the rollout level the effect that the aggregate analyses in Section 6 measure.

H Qualitative Code Rollout Examples

We provide two logged examples from Qwen2.5-Coder-3B-Instruct, where generation begins directly with code rather than an explicit reasoning trace. As in the mathematical examples, each prompt has G=8 rollouts. Under REFT, K=4 first-token branches receive two continuations each. We omit fixed structural prefixes such as code fences and template comments and display each rollout beginning at its first semantic code token. Only representative beginnings are shown, since the purpose is to illustrate how the first-token choice shapes the subsequent program.

H.1 Case 1: Common Substring Detection

Prompt (abridged)

Given two strings s_1 and s_2, determine whether they share a common substring. A substring may consist of a single character. For each of p test cases, output `YES` if the two strings share at least one character and `NO` otherwise.

Standard sampling: $1/8$ full-test pass. Only one of the eight standard rollouts receives full sandbox-test reward. Several failures reach the correct set-intersection idea but break elsewhere in the surrounding program. For example, rollout 3 below replaces input with sys.stdin.read and then attempts to parse the entire input as the test-case count.

Standard rollout 2

```
def twoStrings(s1, s2):
    set1 = set(s1)
    set2 = set(s2)
    if set1 & set2:
        return "YES"
    else:
        return "NO"

def main():
    import sys
    input = sys.stdin.read
    data = input().split()
    p = int(data[0])
```

[...]

Standard rollout 3

```
def twoStrings(s1, s2):
    set1 = set(s1)
    set2 = set(s2)
    if set1.intersection(set2):
        return "YES"
    else:
        return "NO"

import sys
input = sys.stdin.read
p = int(input().strip())
```

[...]

REFT: $6/8$ **full-test pass.** The four selected first semantic tokens are `for`, `s`, `input`, and `T`, at policy ranks 5, 7, 8, and 11, respectively. Six of the eight resulting continuations receive full reward. The successful branches organize the same underlying computation differently from the first line onward.

REFT rollout 1 First Token = `for`

```
for _ in range(int(input())):
    s1 = input()
    s2 = input()
    if len(set(s1) & set(s2)) > 0:
        print("YES")
    else:
        print("NO")
```

REFT rollout 7 First Token = `T`

```
T = int(input().strip())
for _ in range(T):
    s1 = input().strip()
    s2 = input().strip()
    if any(c in s2 for c in s1):
        print("YES")
    else:
        print("NO")
```

The successful branches differ in their input handling and top-level control flow despite originating from extremely low-probability first tokens. Thus, changing the first semantic token affects more than its lexical identity. It changes how the continuation organizes the program.

H.2 Case 2: Counting Zeros in a Binary Representation

Prompt (abridged)

For each input integer N, output the number of zeros in its binary representation. For example, $2 = 10_2$ contains one zero and $4 = 100_2$ contains two.

Standard sampling: $0/8$ **full-test pass.** None of the eight standard rollouts receives full sandbox-test reward. A representative failure applies the right high-level operation but counts zeros directly in Python's `bin()` representation, thereby also counting the zero in its `0b` prefix.

Standard rollout 3

```
def count_zeros_in_binary(n):
    return bin(n).count('0')
```

```
def main():
    import sys
    T = int(sys.stdin.readline())
    for _ in range(T):
        N = int(sys.stdin.readline())
        print(count_zeros_in_binary(N))
```

[...]

REFT: $5/8$ **full-test pass.** The four selected first semantic tokens are `from`, `test`, `input`, and `number`, at policy ranks 3, 11, 13, and 14, respectively. Five of the eight continuations receive full reward. Successful branches recover multiple implementations rather than collapsing onto a single code template.

REFT rollout 2 First Token = `from`

```
from sys import stdin

def count_zeros(n):
    count = 0
    while n > 0:
        if n % 2 == 0:
            count += 1
        n //= 2
    return count
```

[...]

REFT rollout 3 First Token = `test`

```
test_cases = int(input())
for test in range(test_cases):
    N = int(input())
    binary_representation = bin(N)[2:]
    count_of_zeros = binary_representation.count('0')
    print(count_of_zeros)
```

The first token does not determine the algorithm. The two successful branches above develop distinct implementations: one counts zero bits through repeated arithmetic operations, while the other converts the number to a binary string and removes the `0b` prefix before counting. Moreover, even two continuations conditioned on the same `from` first token can differ in correctness. The first token therefore acts as an early routing choice rather than a complete commitment to a particular implementation.

Takeaway. Across both examples, REFT introduces variation before the program has committed to its later syntax, control flow, and implementation details. Low-probability first tokens can seed coherent and successful program trajectories that standard sampling would rarely visit, while all subsequent tokens are generated normally by the policy. In this sense, first-token diversification expands the set of program routes exposed to training without repeatedly perturbing an already formed program.