Reinforcement Learning with Promising Tokens for Large Language Models

Jing-Cheng Pang [1] **Liang Lu** [1] **Xian Tang** [1] **Kun Jiang** [1] **Sijie Wu** [1] **Kai Zhang** [1] **Xubin Li** [1]

Abstract

Reinforcement learning (RL) has become a key tool for improving large language models (LLMs) on reasoning tasks. In practice, LLM rollouts are often generated with truncated decoding rules such as Top-k or Top-p sampling, whereas policy updates are still computed over the full vocabulary. This creates a support mismatch: data are collected from a compact set of plausible tokens, but the optimizer assigns learning signal through logits that were unavailable during rollout. In this work, we first show that successful reasoning trajectories are highly concentrated in the behavior model's high-probability support, suggesting that rollout truncation can preserve most locally useful reasoning actions. We then propose Reinforcement Learning with Promising Tokens (RLPT), a support-aligned RL framework that constructs a state-dependent promising token set during rollout and reuses the same stored support during policy optimization. RLPT computes masked log-probabilities and policy ratios on the rollout support, aligning exploration and optimization on the same action space. Multi-seed experiments on mathematical reasoning benchmarks show consistent gains over GRPO, and ablations confirm that the improvement comes from support alignment rather than Top-k rollout alone.

1. Introduction

Reinforcement Learning (RL) has become a central tool for improving large language models (LLMs) on tasks with verifiable outcomes, such as mathematical reasoning and code generation (Pang et al., 2024; Mu et al., 2024; Shao et al., 2024). By treating an LLM as a policy over next-token actions, RL can optimize sequence-level rewards that are difficult to capture with supervised learning alone.

However, directly applying standard RL algorithms to LLMs exposes an unusually large discrete action space: the full vocabulary, often containing more than 50,000 tokens (Wen et al., 2024). In practice, rollout systems rarely sample freely from this full support. They typically use Top-k (Hopkins et al., 2023), nucleus sampling (Holtzman et al., 2020), or related truncation rules to avoid incoherent low-probability continuations (Sheng et al., 2024). This practical choice creates a subtle mismatch. The behavior policy that collects trajectories is a truncated policy, but the policy update is often computed using log-probabilities and probability ratios over the full vocabulary. As a result, the optimizer may spend learning signal on probability mass that was structurally unavailable during data collection.

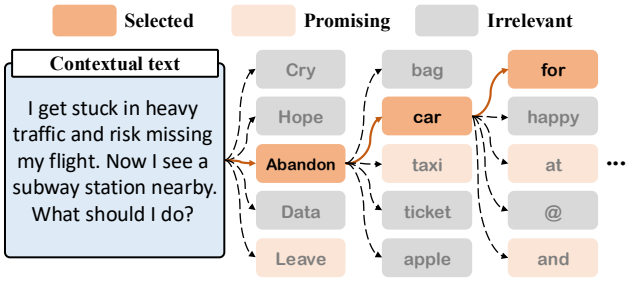

Figure 1. An illustration of decision making over promising tokens. At each step, the behavior policy defines a compact high-likelihood support, and the rollout action is sampled from the renormalized distribution over this support.

This paper studies a simple question: when LLM rollouts are already collected from a truncated token support, should the RL objective be optimized on that same support? We first empirically examine successful reasoning trajectories and find that most tokens used in correct solutions lie within a compact high-probability set under the behavior model. This observation does not imply that tail tokens are never useful. Rather, it clarifies the regime in which practical reasoning RL often operates: truncation can retain most locally useful actions, but once it is used for data collection, the collected trajectory is generated by a policy with a smaller action support than the full-vocabulary policy used in the standard update.

This shifts the focus from choosing a better decoding heuristic to maintaining consistency between data collection and policy optimization. If rollout truncation already defines which actions are available at each state, then evaluating the policy ratio on the full vocabulary introduces off-support

[1]Department of AI Data, Huawei Technologies Co., Ltd.. Correspondence to: Jing-Cheng Pang <pangjc@lamda.nju.edu.cn>.

Published as a paper at the 1st FoGen workshop, ICML 2026, Seoul, South Korea, 2026. Copyright 2026 by the author(s).

competition in the normalization term and assigns gradients to logits that could not have produced the sampled action. Motivated by this view, we propose Reinforcement Learning with Promising Tokens (RLPT), a support-aligned RL framework for LLMs. At each generation step, RLPT constructs a state-dependent promising set from the behavior policy and samples from the renormalized distribution over that set. Crucially, the same mask is stored with the trajectory and reused during policy optimization, so that the probability ratio is computed on the action support that actually generated the data. Thus, RLPT is not intended as a new decoding heuristic; its main role is to align rollout and optimization when a truncated action support is used.

Our contributions are summarized as follows. First, we analyze token ranks in successful reasoning trajectories and show that high-probability supports provide high empirical coverage on math, code, and instruction-following data. Second, we formulate RLPT, a practical support-aligned RL objective that stores behavior-policy masks during rollout and reuses them in the policy update. Third, we provide a variance-oriented analysis explaining which gradient components are removed by support restriction, while explicitly discussing the assumptions and limitations of fixed-size masks. Finally, multi-seed experiments and controlled ablations show that RLPT improves GRPO on standard mathematical reasoning benchmarks and that the gain is primarily due to consistent masking across rollout and optimization rather than rollout truncation alone.

2. Preliminary

RL for LLM optimization. We consider a standard language modeling task in which an LLM serves as a policy π_θ parameterized by θ. Given a context input (prompt or question) x, the model generates a response y composed of a sequence of tokens $y = (y_1, y_2, \ldots, y_T)$, where each token y_t belongs to a fixed vocabulary space $\mathcal{V}$. The probability of generating the entire sequence is given by the chain rule: $\pi_\theta(y \mid x) = \prod_{t=1}^{T} \pi_\theta(y_t \mid x, y_{<t})$, where $y_{<t}$ denotes the partial sequence generated prior to step t.

To align the LLM with human preferences or specific logical rules, we formulate the generation process as a Markov Decision Process (MDP (Puterman, 1994)) defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$:

- State space $\mathcal{S}$: The state s_t at time step t consists of the prompt and the history of generated tokens, i.e., $s_t = (x, y_1, \ldots, y_{t-1})$;
- Action space $\mathcal{A}$: the space of the tokens $\mathcal{V}$;
- Transition $\mathcal{T}$: $s_{t+1} = s_t \cup \{a_t\}$;
- Reward function $\mathcal{R}$: a score $r(s_t, a_t)$ that reflects the quality of the generated answer to the question, typically given at the end of the sequence (e.g., $r_T = R(x, y)$) based on correctness verification, with intermediate rewards being zero;
- Discount factor γ: We typically set $\gamma = 1$ for episodic generation tasks.

The goal of RL training is to find an optimal policy π_θ that maximizes the expected cumulative reward over the prompt distribution $\mathcal{D}$:

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi_\theta(\cdot|x)}[R(x, y)],$$

where $R(x, y)$ represents the reward (e.g., answer correctness) received at the end of the episode.

3. Analyzing the Support of Successful Reasoning Trajectories

Before introducing the method, we examine the empirical premise behind support-aligned optimization: *when a model produces a successful reasoning trajectory, how often are the generated tokens already contained in a compact high-probability support?* This question is different from asking whether tail tokens are never useful. Instead, it measures whether a truncated support can preserve most of the actions used by successful trajectories in the reasoning settings we study. We conduct this analysis using Qwen-series models (Bai et al., 2023). We first define the promising token set.

Definition 3.1 (**Promising Tokens**). *Given the current state s_t and the policy distribution $\pi(\cdot|s_t)$, let rank$(v \mid s_t)$ denote the rank of a token $v \in \mathcal{V}$ when sorted in descending order of its probability. The set of promising tokens, denoted as $\mathcal{P}_t$, is defined as the top-K candidates:*

$$\mathcal{P}_t = \{v \in \mathcal{V} \mid \textit{rank}(v \mid s_t) \leq K\}. \tag{1}$$

Consequently, $|\mathcal{P}_t| = K$ and $\mathcal{P}_t \subset \mathcal{V}$.

This definition constructs a compact, state-dependent action support. To verify the coverage of $\mathcal{P}_t$, we analyze the token ranks of *successful trajectories* from two perspectives: labeled solutions and model-generated correct solutions.

Setup. We employ `Qwen3-8B` and `Qwen3-32B` as base models. We use three diverse datasets: GSM8K (Cobbe et al., 2021) for mathematics, HumanEval (Chen, 2021) for coding, and AlpacaEval (Taori et al., 2023) for general instruction following. For each step t in a successful trajectory y, we compute the rank of the target token y_t within the base model's predicted distribution $\pi_{\text{base}}(\cdot|y_{<t})$. We define the *top-$K$ coverage rate* as the percentage of tokens in these trajectories that satisfy $y_t \in \mathcal{P}_t$.

3.1. Analysis of Ground-truth Solution

Observation 1: ***Labeled solutions have high, but not perfect, coverage under compact token supports.***

Table 1. Top-K coverage rate (%) of successful trajectory tokens from labeled solutions. Math, Code, and General denote GSM8K, HumanEval, and AlpacaEval dataset, respectively.

Metric	Qwen3-8B			Qwen3-32B		
	Math	Code	General	Math	Code	General
Top-2	91.5	92.0	83.0	92.1	92.7	83.8
Top-4	95.3	95.0	91.0	96.1	95.3	91.7
Top-8	97.4	96.5	95.4	98.1	96.7	95.9
Top-16	98.6	97.2	97.6	99.0	97.5	98.0
Top-32	99.2	97.7	98.8	99.5	98.0	99.0

We first investigate whether labeled solutions fall within the promising tokens predicted by the base model. As shown in Tab. 1, solution tokens are strongly concentrated in the head of the distribution. For `Qwen3-32B`, over 99.5% of the labeled math tokens fall within the Top-32 candidates. Even for the smaller 8B model, Top-32 coverage exceeds 97% across all three domains. This result supports the feasibility of compact supports, but it should not be read as a hard guarantee. The remaining out-of-support tokens can matter in some domains, especially when rare symbols, library names, or unusual formatting are required. Therefore, RLPT treats the promising set as an optimization support for reasoning-oriented RL, not as a universal proof that tail tokens are dispensable.

3.2. Analysis of Model-generated Solution

Observation 2: ***Model-generated correct solutions are even more concentrated in the high-probability support.***

While the previous experiment studies labeled solutions, one might worry that the model needs to explore low-probability alternatives to find its own correct solution path. To address this, we sample multiple answers from the base model and analyze only those that reach the correct final answer. Tab. 2 reports the rank of tokens in these *model-generated successful paths*. The results show that for Math and Code tasks, 100% of the tokens in correct solutions are located within the Top-8 candidates. This indicates that when the model successfully reasons in our evaluated settings, it relies mostly on high-likelihood tokens. Consequently, the challenge of RL may often be less about searching the entire vocabulary and more about improving the relative preference among plausible next moves.

Table 2. Top-K coverage rate (%) of successful trajectory tokens from model-generated solution.

Metric	Qwen3-8B			Qwen3-32B		
	Math	Code	General	Math	Code	General
Top-2	98.9	98.1	93.2	92.1	97.8	92.4
Top-4	99.9	99.6	98.0	100	100	100
Top-8	100	100	100	100	100	100

3.3. Coverage on Harder Mathematical Data

The original coverage analysis above includes GSM8K, HumanEval, and AlpacaEval. To test whether the concentration pattern persists on harder mathematical data, we additionally evaluate `Qwen3-8B` on Math-17k. As shown in Tab. 3, the Top-4 support covers 98.15% of successful trajectory tokens and Top-16 covers 99.77%. This supports using a small K for mathematical reasoning, while also motivating the ablation over K in Appendix C.4.

Table 3. Top-K coverage rate (%) of successful trajectory tokens on Math-17k.

Model	Top-2	Top-4	Top-8	Top-16
Qwen3-8B	95.32	98.15	99.34	99.77

4. Method

In this section, we present RLPT, a support-aligned RL framework for LLMs. The method has two steps, as shown in Fig. 2: first, construct a binary mask M_t from the behavior policy at each rollout state; second, reuse the same stored mask when evaluating the current policy during optimization. The purpose of this design is to keep the data-collection support and the policy-update support consistent.

4.1. Promising Token Construction

The foundation of RLPT is the identification of *promising tokens*. At each time step t, given the current state s_t, we use the behavior policy π_{old} (the policy used for data collection) to compute the probability distribution over the vocabulary $\mathcal{V}$.

We formally define the *Promising Mask* $M_t \in \{0,1\}^{|\mathcal{V}|}$ as a binary vector over the vocabulary. Let $\mathcal{P}_t$ be the set of top-K tokens determined by $\pi_{\text{old}}(\cdot|s_t)$. The elements of the mask vector M_t are defined as:

$$M_t[v] = \mathbb{I}(v \in \mathcal{P}_t) = \begin{cases} 1, & \text{if } v \in \mathcal{P}_t \\ 0, & \text{otherwise} \end{cases}, \tag{2}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Fixed Top-K is used as the default construction because it gives a controlled way to study support restriction. The framework itself only requires that the rollout support can be stored and reused during optimization; adaptive supports such as Top-p can also be plugged into the same objective.

4.2. Policy Rollout and Optimization with Promising Token Masking

RLPT integrates the promising mask into the standard RL pipeline by modifying both the rollout and the policy update.

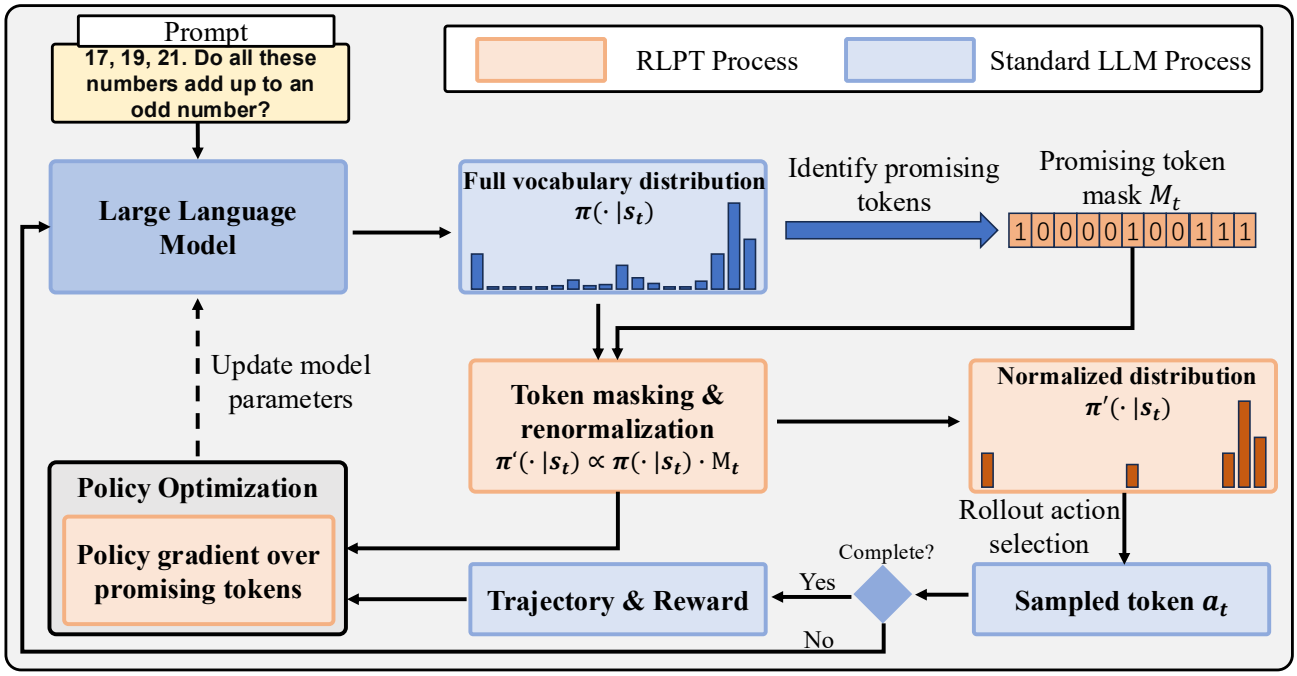

Figure 2. The overall framework of RLPT. The behavior policy defines a promising token support during rollout, and the stored support is reused when computing masked log-probabilities and policy ratios during optimization.

Policy Rollout. During data generation, we sample from a masked behavior distribution $\tilde{\pi}_{\text{old}}$ rather than from the full-vocabulary distribution. The probability of selecting a token a_t is renormalized over the promising set:

$$\tilde{\pi}_{\text{old}}(a_t \mid s_t) = \frac{\pi_{\text{old}}(a_t \mid s_t) \cdot M_t[a_t]}{\sum_{v \in \mathcal{V}} \pi_{\text{old}}(v \mid s_t) \cdot M_t[v]}. \tag{3}$$

This operation is equivalent to Top-K sampling when $\mathcal{P}_t$ is defined by Top-K. Crucially, RLPT stores M_t together with the trajectory, so the optimization step can evaluate the current policy on the same support that was available to the behavior policy.

Policy Optimization. A key difference between RLPT and rollout-only truncation is consistent masking during optimization. Standard implementations may collect trajectories from a truncated behavior policy while computing the policy ratio using full-vocabulary log-probabilities. RLPT instead evaluates the current policy π_θ on the same stored support.

During the backward pass, we apply the pre-computed mask M_t (derived from the behavior policy) to the current policy's logits. The masked probability is:

$$\tilde{\pi}_\theta(a_t \mid s_t) = \text{Softmax}\left(\text{Logits}_\theta(s_t) + (1 - M_t) \cdot (-\infty)\right)_{a_t}. \tag{4}$$

The mask is not recomputed from the evolving current policy within the same optimization batch. Recomputing it could assign zero support to an action that was sampled during rollout. Reusing the behavior-policy mask keeps every sampled action inside the optimization support.

4.3. Integration with RL Algorithms

RLPT is algorithm-agnostic and can be integrated with common RL objectives. We illustrate this using Group Relative Policy Optimization (GRPO) (Shao et al., 2024) as a representative baseline. Given a group of trajectories $Traj_1, \ldots, Traj_G$ sampled with the masked policy, the objective function of RLPT-GRPO is formulated as:

$$\begin{aligned}\mathcal{J}_{\text{RLPT}}(\theta) = &\mathbb{E}_{Traj \sim \tilde{\pi}_{\text{old}}}\Bigg[\frac{1}{T}\sum_{t=1}^{T} \min\Bigg(\frac{\tilde{\pi}_\theta(a_t|s_t)}{\tilde{\pi}_{\text{old}}(a_t|s_t)} A_t, \\ &\text{clip}\left(\frac{\tilde{\pi}_\theta(a_t|s_t)}{\tilde{\pi}_{\text{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right) A_t\Bigg)\Bigg],\end{aligned} \tag{5}$$

where A_t is the advantage computed from rewards. The key difference from standard GRPO is that the probability ratio $\frac{\tilde{\pi}_\theta}{\tilde{\pi}_{\text{old}}}$ is computed within the stored support defined by M_t. This aligns the policy improvement step with the support used for exploration.

4.4. Theoretical Justification

In this subsection, we analyze how support restriction changes the gradient estimator. The goal is not to claim a universal variance bound for all settings, since renormalization over $\mathcal{P}_t$ also changes the distribution over unmasked tokens. Instead, we isolate the tail-logit components that are

removed when the policy is optimized on the same restricted support used for rollout.

Consider the gradient of the objective function $J(\theta)$ with respect to the logits $z_t \in \mathbb{R}^{|\mathcal{V}|}$ of the policy at step t. The standard policy gradient estimator $\hat{g}$ can be expressed as:

$$\hat{g} = \nabla_{z_t} \log \pi_\theta(a_t|s_t) \cdot A_t, \tag{6}$$

where A_t is the advantage function. For the softmax parameterization $\pi(a|s) = \frac{e^{z_a}}{\sum_{v\in\mathcal{V}} e^{z_v}}$, the gradient of the log-probability with respect to the logit vector z is given by $\nabla_z \log \pi(a|s) = \mathbf{e}_a - \pi$, where $\mathbf{e}_a$ is the one-hot vector for action a and π is the probability vector over $\mathcal{V}$.

Let $\mathcal{T} = \mathcal{V} \setminus \mathcal{P}_t$ denote the set of "tail" tokens that are masked out in RLPT. We analyze the variance of the gradient estimator by decomposing the contribution of the promising set $\mathcal{P}_t$ and the tail set $\mathcal{T}$.

Proposition 4.1 (Tail-gradient Removal). *For a fixed mask $\mathcal{P}_t$ and bounded advantage A_t, the masked policy gradient has zero logit-gradient components for all tail tokens in $\mathcal{T} = \mathcal{V} \setminus \mathcal{P}_t$. Therefore, the variance associated with these tail-logit components is removed relative to a full-vocabulary softmax estimator.*

We present the proof of Prop. 4.1 in Appendix D.1.

This analysis explains why RLPT can stabilize training when tail-token gradients are mostly noise. In Sec. 6.4, we complement this argument with empirical gradient statistics and controlled ablations.

5. Related Work

In this section, we review related work from the following areas.

5.1. Reinforcement Learning for LLM Training

Reinforcement learning (RL) has become a common approach for aligning Large Language Models (LLMs) with human intent and improving reasoning capabilities. Early works, such as RLHF (Ouyang et al., 2022), employ PPO (Schulman et al., 2017) to optimize policies with respect to learned reward models. DPO (Rafailov et al., 2023) and its variants (Azar et al., 2024; Ethayarajh et al., 2024) simplify preference optimization by avoiding online sampling, while outcome-based RL methods such as GRPO (Shao et al., 2024) and RFT (Trung et al., 2024) improve mathematical and code reasoning with verifiable rewards. Recent work also studies how to avoid the inefficiency of token-level RL from other angles. DeepLatent Reasoning shifts exploration from token sequences to latent reasoning-chain encodings (Shan et al., 2026), while spatio-temporal pruning improves RL for diffusion language models by constraining exploration and removing redundant refinement steps (Liu et al., 2026a). These methods share the motivation of improving efficiency and stability under high-dimensional generation, but they change the representation or generation process. RLPT instead keeps the standard autoregressive policy and aligns the token support used by rollout and optimization.

5.2. Decoding-time Action Pruning Strategies

Managing the vast action space of LLMs is widely studied during inference. Deterministic strategies, such as greedy decoding, can yield repetitive loops (Pipis et al., 2025), while stochastic methods such as Top-k sampling (Hopkins et al., 2023), nucleus sampling (Holtzman et al., 2020), and Min-p sampling (Nguyen et al., 2024) truncate the tail of the probability distribution to improve generation quality. RLPT is closely related to these methods, but addresses a different stage of the pipeline. Top-k and Top-p specify how to sample tokens during rollout. RLPT asks whether the same truncated support should also define the policy distribution used for optimization. In this sense, RLPT does not compete with decoding heuristics; it turns a chosen rollout support into a support-aligned RL objective.

5.3. Token-Level Stabilization in LLM RL

Several recent studies identify token-level effects as a source of instability in LLM RL. Most closely related to our motivation, STAPO (Liu et al., 2026b) argues that a small fraction of rare spurious tokens can dominate token-wise policy gradients when they inherit sequence-level rewards, and suppresses their gradient perturbations during optimization. RLPT is complementary: rather than detecting problematic tokens after trajectories are generated, it defines a behavior-policy support during rollout and evaluates the policy ratio within the same stored support. VAM (Zhang et al., 2026) studies action masking for controllable exploration in chess-based RL post-training by verbalizing a valid action set in the prompt and enforcing outputs from that set. In contrast, RLPT does not require an externally supplied legal-action set or prompt-level verbalization; it derives a state-dependent support from the model distribution itself and applies it directly in the optimization objective.

5.4. Reinforcement Learning with Large Action Space

Handling large or continuous action spaces is a longstanding challenge in RL research. Wolpertinger-style methods (Zhong et al., 2018) embed actions in a continuous space to handle large discrete spaces, while action decomposition for LLM agents (Wen et al., 2024) improves credit assignment by factorizing complex token-level decisions. Invalid-action masking (Huang & Ontañón, 2022) is another relevant line of work: it removes actions that are known to be illegal before sampling and gradient computation. RLPT shares the same high-level idea of reducing the effective action

support, but differs in how the mask is obtained and used. Invalid-action masking usually relies on an external rule or learned validity model; RLPT uses the behavior policy's own distribution to define a state-dependent support and stores that support for the policy update. MSL (Wang et al., 2025) also masks tokens for efficient recommendation tuning, but relies on a task-specific filter. RLPT is designed for open-vocabulary reasoning settings where explicit validity rules are typically unavailable.

6. Experiment

In this section, we evaluate the effectiveness of RLPT. The main goal of our experiments is to answer the following key research questions: (1) Does support-aligned optimization improve mathematical reasoning accuracy? (Sec. 6.2) (2) Are the gains caused by support alignment rather than rollout truncation alone? (Sec. 6.3) (3) How does RLPT affect training stability? (Sec. 6.4) and (4) how do selector design and support size affect performance? (Appendix C.4) We begin by introducing the experimental setting.

6.1. Experimental Setting

Datasets for evaluation. To verify the effectiveness of RLPT, we evaluate our approach on benchmarks covering three representative domains:

- **Mathematical reasoning:** We utilize Math-17k (Yu et al., 2025), AIME-24, AIME-25, and MATH500 (Hendrycks et al., 2021) to evaluate multi-step logical deduction. These tasks are the main focus of our evaluation because reasoning accuracy depends strongly on selecting correct intermediate steps.
- **Code:** We leverage OpenR1-Code (Hugging Face, 2025) to evaluate proficiency in algorithmic synthesis. Unlike natural language, code generation necessitates navigating deterministic syntax and long-range functional dependencies, providing a benchmark for the model's structural coherence.
- **Telecom:** We also consider telecommunication tasks to verify broader application of RLPT method, including Datacom and Wireless. We construct a dataset for each task comprising 12,000 question-answer pairs, synthesized from a diverse corpus of product documentation, technical solutions, and domain knowledge bases. The datasets encompass diverse question types, including single-choice, multiple-choice, and open-ended QA, covering fundamental principles, product concepts, terminology understanding, and multi-step reasoning tasks.

Appendix E.1 shows examples of the input and output of these datasets.

Baseline RL methods. In our experiments, we use Group Relative Policy Optimization (**GRPO** (Shao et al., 2024)) as the primary baseline because it is widely used for rule-verifiable reasoning. We also include controlled comparisons with Decoupled Clip and Dynamic Sampling Policy Optimization (**DAPO** (Yu et al., 2025)) to test whether RLPT can be integrated with another RL objective. GRPO eliminates the critic model by estimating advantages through group-wise normalization of rewards. DAPO builds on this framework by introducing dynamic sampling and decoupled clipping.

Implementation details. Experiments are implemented using the MindSpeed-RL framework (Feng et al., 2025). We primarily employ Qwen3-8B as the base model and additionally test Qwen3-4B and Llama-3.1-8B-Instruct for model-generalization analysis. Reward signals are provided by rule-based verifiers for Math, execution sandboxes for Code, and model-based evaluation (Deepseek-V3 (DeepSeek-AI, 2024)) for Telecom. Unless stated otherwise, we use a promising set size of $K = 4$ and report results at 200 training steps. For the main mathematical results, we report mean and standard deviation over 5 independent runs. Computing resources includes clusters with 256 KUNPENG 920 CPUs and 8 Ascend 910B3 NPUs. Full hyperparameters are in Appendix E.3. We refer the readers to Appendix E for more details about the experimental setting.

6.2. Main Results

Performance on mathematical tasks. Tab. 4 presents the main results on Qwen3-8B. RLPT improves GRPO on all four mathematical benchmarks. On Math-17k, GRPO+RLPT achieves $37.78\% \pm 0.61$, compared with $35.78\% \pm 1.36$ for GRPO. On AIME24 and AIME25, RLPT improves the mean accuracy by 1.66% and 2.20%, respectively. On MATH500, a standard held-out benchmark, RLPT improves GRPO from $56.64\% \pm 0.88$ to $60.12\% \pm 1.78$. These results suggest that support-aligned optimization provides a consistent benefit in the reasoning setting where compact high-probability supports have high empirical coverage.

Scope beyond mathematics. We also evaluate code and telecom tasks as auxiliary settings, but we treat these results as preliminary because several gains are small and some are mixed. The clearest evidence for RLPT is currently on mathematical reasoning; broader conclusions for open-ended or domain-heavy generation require more extensive evaluation.

Training Efficiency. As visualized in Fig. 3, RLPT achieves higher reward in the early stages compared to GRPO. This is consistent with the support-alignment view: when rollout already uses a compact support, optimizing the policy ratio on that same support can concentrate updates on the choices that were actually available during trajectory generation.

Table 4. Main mathematical reasoning results. We report mean accuracy (%) ± standard deviation over 5 independent runs when available. MATH500 is evaluated as a standard held-out subset of the MATH benchmark.

Method	Math-17k	AIME24	AIME25	MATH500
No Training	24.96 ± 0.59	20.14 ± 1.68	16.53 ± 2.29	55.48 ± 0.48
GRPO	35.78 ± 1.36	21.57 ± 2.01	19.73 ± 1.35	56.64 ± 0.88
GRPO+RLPT	$\mathbf{37.78 \pm 0.61}$	$\mathbf{23.23 \pm 1.95}$	$\mathbf{21.93 \pm 1.04}$	$\mathbf{60.12 \pm 1.78}$

Table 5. Auxiliary single-run results on code and telecom tasks. These results are included to show compatibility beyond math, but we do not use them as the main evidence for broad domain generalization.

Method	Math			Telecom		Code
	Math-17k	AIME24	AIME25	Datacom	Wireless	OpenR1-Code
GRPO	34.7	20.0	19.3	50.87	52.17	40.18
GRPO+RLPT	**38.3**	**23.3**	18.0	**51.30**	**55.65**	**40.92**
DAPO	36.4	**20.7**	17.3	54.43	49.57	**39.87**
DAPO+RLPT	**39.7**	19.3	**20.7**	**54.78**	**50.43**	39.01

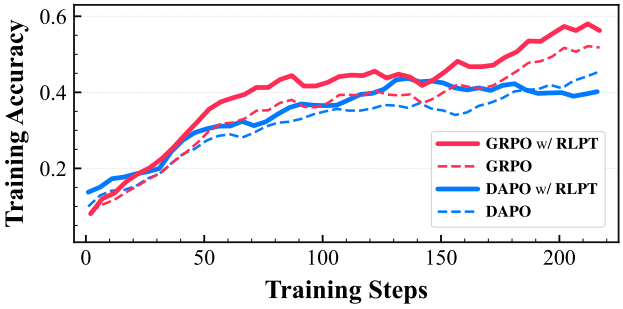

Figure 3. Training curves on Math-17k dataset.

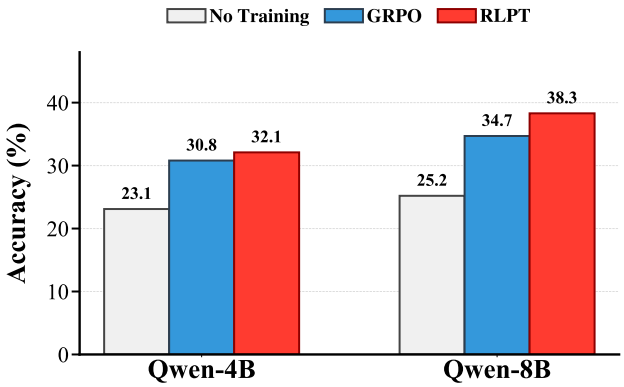

Figure 4. Performance of RLPT method on different sizes of models.

Model generalization. To verify that the effect is not confined to a single backbone, we evaluate RLPT on different model families. Fig. 4 reports the Qwen3-4B and Qwen3-8B comparison, and Tab. 6 reports an additional Llama-3.1-8B-Instruct experiment on Math-17k. RLPT improves Llama-3.1-8B-Instruct from 33.2% to 35.4%, suggesting that support alignment can transfer beyond the Qwen series.

Table 6. Model-family generalization on Math-17k with Llama-3.1-8B-Instruct.

Method	Math-17k Acc.
GRPO	33.2
GRPO+RLPT	**35.4**

6.3. Ablation on Support Alignment

We next isolate whether the improvement comes from Top-K rollout alone or from using the same support during both rollout and optimization. Tab. 7 shows that rollout truncation without optimization masking is weaker than full RLPT, and optimization-only masking performs poorly because the update support is no longer guaranteed to match the sampled actions. This supports the central claim that the key design choice is support alignment, not merely choosing a smaller rollout distribution.

Table 7. Controlled ablation on Math-17k. RLPT uses the same stored Top-K mask in both rollout and optimization.

Rollout	Optimization Mask	Acc.
Top-K	None	36.9
Top-p	None	37.7
Full	Top-K	31.7
Top-K	Same stored Top-K	**38.3**

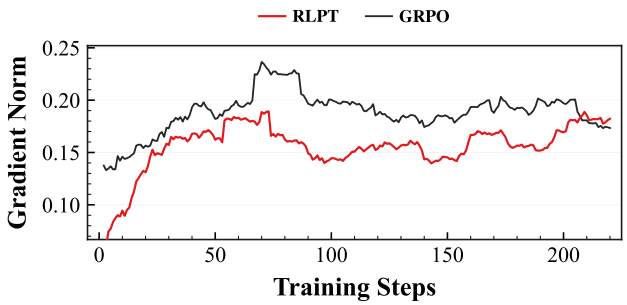

Figure 5. Gradient norm curves of GRPO and GRPO+RLPT during the training process.

We also compare different support-construction rules inside the same support-aligned framework. As shown in Tab. 8, fixed Top-K outperforms Top-p in our current mathematical reasoning setup. This result does not imply that Top-K is universally optimal; rather, it shows that adaptive supports require careful tuning, especially when overly large supports reintroduce low-probability tail tokens.

Table 8. Support construction variants for RLPT.

Support Rule	Math-17k	AIME25	AIME24
Top-p ($p = 0.95$)	35.6	16.7	22.4
Top-K ($K = 4$)	**38.3**	**19.5**	**22.5**

6.4. Performance Analysis of RLPT Method

To better understand why RLPT improves training, we analyze gradient statistics and qualitative decision-making.

Gradient norm during the training process. The analysis in Sec. 4 predicts that support restriction removes gradient components associated with masked tail logits. We therefore monitor the norm of the policy gradient during training. Fig. 5 shows that GRPO+RLPT has a smoother gradient-norm trajectory than GRPO. Across the monitored training process, the standard deviation of the gradient norm decreases from 0.0032 for GRPO to 0.0029 for GRPO+RLPT, a relative reduction of about 9.4%. We view this as diagnostic evidence for more stable updates, not as a complete proof that total estimator variance is lower in every setting.

Case study on decision making with promising token. To inspect what the promising support contains, we examine specific inference steps from the GSM8K dataset (Cobbe et al., 2021) using the trained model. Tab. 9 illustrates representative cases. Given a context requiring a reasoning step, the promising set contains plausible continuations and logical connectors, e.g., so, since, but. This observation is consistent with the view that the high-probability support often contains the locally plausible reasoning choices, while the optimization objective is responsible for shifting preference among them.

Table 9. Case study on the decision making over promising tokens.

Contextual text	Promising tokens
A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take? Answer: It takes 2/2=1 bolt of white fiber.	[Total, It, So, The, But, A, That, In]
Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals … 60 cups of feed per day.	[She, In, Since, Total, So, They, Thus, Each]

6.5. Additional Diagnostics

Appendix C.4 provides additional diagnostic experiments, including a comparison with an explicit selector policy and an ablation over the promising-set size K. These results further support the choice of deriving the support from the behavior policy and using a compact support in the main experiments.

7. Conclusion & Future Work

In this work, we proposed RLPT, a support-aligned RL framework for LLM reasoning. Instead of treating token truncation only as a rollout heuristic, RLPT stores the behavior-policy support used during rollout and reuses it during policy optimization. This aligns the action support used for data collection with the support used in the policy ratio. Our empirical analysis shows that successful reasoning trajectories are highly concentrated in compact high-probability supports, and the results show consistent improvements over GRPO on mathematical reasoning benchmarks. However, there are still some limitations. First, fixed Top-K supports may be sub-optimal, as adaptive thresholding methods such as Top-p or Min-p sampling (Nguyen et al., 2024) may be preferable when uncertainty varies strongly across steps. Second, restricting the action support can exclude critical low-probability tokens, especially for weak base models, rare symbolic reasoning, library-heavy code generation, or open-ended creative tasks. This may lead to the missing of the correct answers. Third, the experimental evidence is currently on mathematical reasoning. The code and telecom results are encouraging but mixed, and broader claims require more extensive multi-seed evaluation. Future work should study adaptive support construction, failure detection when coverage is low, and integration with complementary methods such as action decomposition.

Impact Statements

This paper presents work whose goal is to advance the field of machine learning, specifically in making reinforcement learning for large language models more efficient and stable. By improving the sample efficacy and training robustness of RL alignment techniques, RLPT could contribute to the development of more capable and reliable AI systems. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Azar, M. G., Guo, Z. D., Piot, B., Munos, R., Rowland, M., Valko, M., and Calandriello, D. A general theoretical paradigm to understand learning from human preferences. In *AISTATS*, 2024.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., et al. Qwen technical report. *arXiv preprint*, abs/2309.16609, 2023.

Chen, M. Evaluating large language models trained on code. *arXiv preprint*, abs/2107.03374, 2021.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint*, abs/2110.14168, 2021.

DeepSeek-AI. Deepseek-v3 technical report. *arXiv*, abs/2412.19437, 2024.

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. KTO: model alignment as prospect theoretic optimization. *arXiv preprint*, abs/2402.01306, 2024.

Feng, L., Pan, C., Guo, X., Mei, F., Ning, B., Zhang, J., Liu, X., Zhou, B., Shu, Z., Liu, C., Yang, G., Han, Z., Wang, J., and Wang, B. Mindspeed RL: distributed dataflow for scalable and efficient RL training on ascend NPU cluster. *arXiv*, abs/2507.19017, 2025.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks*, 2021.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration. In *ICLR*, 2020.

Hopkins, A. K., Renda, A., and Carbin, M. Can llms generate random numbers? evaluating llm sampling in controlled domains. In *ICML SODS Workshop*, 2023.

Huang, S. and Ontañón, S. A closer look at invalid action masking in policy gradient algorithms. In *FLAIRS*, 2022.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

Liu, J., Wang, X., Zhong, Y., Lian, D., and Yang, Y. Efficient and stable reinforcement learning for diffusion language models. *arXiv preprint*, abs/2602.08905, 2026a.

Liu, S., He, Z., Zhan, G., Tao, L., Zheng, Z., Wu, J., Wang, Y., Guan, Y., Sheng, K., Zhang, B., Li, K., Duan, J., and Li, S. E. STAPO: Stabilizing reinforcement learning for LLMs by silencing rare spurious tokens. *arXiv preprint*, abs/2602.15620, 2026b.

Mu, T., Helyar, A., Heidecke, J., Achiam, J., Vallone, A., Kivlichan, I., Lin, M., Beutel, A., Schulman, J., and Weng, L. Rule based rewards for language model safety. In *NeurIPS*, 2024.

Nguyen, M. N., Baker, A., Neo, C., Roush, A., Kirsch, A., and Shwartz-Ziv, R. Turning up the heat: Min-p sampling for creative and coherent llm outputs. *arXiv preprint*, abs/2407.01082, 2024.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P. F., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Pang, J., Wang, P., Li, K., Chen, X., Xu, J., Zhang, Z., and Yu, Y. Language model self-improvement by reinforcement learning contemplation. In *ICLR*, 2024.

Pipis, C., Garg, S., Kontonis, V., Shrivastava, V., Krishnamurthy, A., and Papailiopoulos, D. Wait, wait, wait... why do reasoning models loop? *arXiv preprint*, abs/2512.12895, 2025.

Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

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

Shan, L., Chen, H., Wang, Y., Liu, Z., and Li, W. Latent-space contrastive reinforcement learning for stable and efficient LLM reasoning. *arXiv preprint*, abs/2601.17275, 2026.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint*, abs/2402.03300, 2024.

Sheng, G., Zhang, C., Ye, Z., Wu, X., Zhang, W., Zhang, R., Peng, Y., Lin, H., and Wu, C. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint*, abs/2409.19256, 2024.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*, 3(6):7, 2023.

Trung, L. Q., Zhang, X., Jie, Z., Sun, P., Jin, X., and Li, H. Reft: Reasoning with reinforced fine-tuning. In *ACL*, 2024.

Wang, B., Liu, F., Chen, J., Lou, X., Zhang, C., Wang, J., Sun, Y., Feng, Y., Chen, C., and Wang, C. MSL: not all tokens are what you need for tuning LLM as a recommender. In *SIGIR*, 2025.

Wen, M., Wan, Z., Wang, J., Zhang, W., and Wen, Y. Reinforcing LLM agents via policy optimization with action decomposition. In *NeurIPS*, 2024.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., et al. DAPO: An open-source llm reinforcement learning system at scale. *arXiv preprint*, abs/2503.14476, 2025.

Zhang, Z., Wang, Z., Du, Y., and Fang, F. VAM: Verbalized action masking for controllable exploration in RL post-training – a chess case study. *arXiv preprint*, abs/2602.16833, 2026.

Zhong, C., Gursoy, M. C., and Velipasalar, S. A deep reinforcement learning-based framework for content caching. In *CISS*, 2018.

Appendix

A. Discussion

A.1. Support Mismatch in LLM Optimization with Truncated Rollouts

We discuss a support mismatch that can arise when LLM RL systems use truncated decoding during rollout. Standard RL approaches for LLMs (e.g., PPO) often employ decoding strategies such as Top-k or Nucleus (Top-p) sampling during data collection to avoid incoherent text from the long tail. Let π_θ denote the parameterized policy over the full vocabulary $\mathcal{V}$. The actual behavior policy π_b during rollout is a truncated version:

$$\pi_b(a_t|s_t) = \text{Truncate}(\pi_\theta(a_t|s_t); k, p),$$

where the support of π_b is strictly smaller than $\mathcal{V}$. However, during the optimization phase, standard baselines typically calculate policy gradients (and KL-divergence penalties) based on the original distribution π_θ over the full vocabulary.

This discrepancy is not the same as ordinary PPO policy lag, which is handled by importance ratios. The issue is that the rollout support and the optimization support differ. RLPT addresses this by integrating the rollout mask into the policy definition used for optimization. By optimizing over the same masked support used for generation, the policy ratio is evaluated on the action space that actually produced the data.

A.2. Practical Scope and Failure Modes

RLPT is designed for reasoning-oriented RL in which the base model already assigns non-negligible probability to useful next-step continuations. This is the regime suggested by the coverage analysis in Sec. 3. The method is less suitable when the desired action is systematically outside the high-probability support. We summarize the main failure modes and practical mitigations in Tab. 10.

Table 10. Potential failure modes of RLPT and practical mitigations.

Setting	**Risk**	**Possible Mitigation**
Weak base model on very hard tasks	Correct continuations may be absent from the Top-K support, making the mask overly restrictive.	Increase K, use an adaptive support rule, or delay masking until the model reaches sufficient coverage.
Rare-symbol math or library-heavy code	Important low-frequency symbols, API names, or operators may be filtered out.	Use larger supports for symbol-heavy domains, whitelist task-specific symbols when available, or combine Top-K with Top-p.
Open-ended generation	A tight support can reduce linguistic diversity and produce rigid outputs.	Use larger K, Top-p/Min-p supports, or apply RLPT only to verifiable reasoning segments.
Long optimization over stale masks	A mask stored from the behavior policy may become less representative if too many update epochs are taken.	Refresh masks whenever a new rollout batch is collected and limit the number of update epochs per batch.

B. Algorithm and Implementation Details

Algorithm 1 RLPT-GRPO with Stored Promising Masks

Input: Initial policy π_θ, prompt distribution $\mathcal{D}$, group size G, promising set size K, clip ratio ϵ
for each training iteration **do**
  Sample a batch of prompts $x \sim \mathcal{D}$
  for each prompt x and rollout $g \in \{1, \ldots, G\}$ **do**
    Initialize state $s_1 = x$
    for $t = 1, \ldots, T$ **do**
      Compute behavior distribution $\pi_{\text{old}}(\cdot|s_t)$
      Construct $\mathcal{P}_t = \text{TopK}(\pi_{\text{old}}(\cdot|s_t), K)$ and mask M_t
      Sample $a_t \sim \tilde{\pi}_{\text{old}}(\cdot|s_t)$, where $\tilde{\pi}_{\text{old}}$ is renormalized over $\mathcal{P}_t$
      Store $(s_t, a_t, M_t, \tilde{\pi}_{\text{old}}(a_t|s_t))$ and update $s_{t+1} = (s_t, a_t)$
    end for
    Evaluate the terminal reward $R(x, y^{(g)})$
  end for
  Compute group-normalized advantages A_t following GRPO
  for each optimization minibatch **do**
    Recompute current logits and apply the stored mask M_t
    Compute masked probability $\tilde{\pi}_\theta(a_t|s_t)$
    Update θ with the clipped objective in Eq. (4)
  end for
end for

B.1. Computational Cost and Storage

RLPT is lightweight in the sense that it does not introduce an auxiliary model, extra supervision, or a separate training stage. It should not be interpreted as a wall-clock acceleration method. The main additional operations are support selection and mask storage. Since LLM generation already computes full-vocabulary logits before sampling, Top-K selection can be implemented with standard partial selection kernels. During optimization, applying the stored mask is a logit-level operation before the softmax. Tab. 11 summarizes the practical overhead.

Table 11. Practical cost breakdown of RLPT relative to standard GRPO.

Component	**Additional Cost**	**Implementation Note**
Rollout forward pass	No additional model forward pass.	The full-vocabulary logits are already computed by the policy.
Support selection	Top-K or Top-p selection per decoding step.	Comparable to standard truncated decoding used in rollout systems.
Stored information	Token indices of the promising set or an equivalent sparse mask.	For fixed K, store K token ids per generated token instead of a dense vocabulary mask.
Optimization forward pass	Mask application before softmax.	The current policy is evaluated on stored supports; no auxiliary selector is needed.
Memory overhead	$O(TK)$ token ids per trajectory.	With $K = 4$, this is small compared with activations and sequence tokens in RL training.

C. Additional Experimental Details

C.1. Dataset and Evaluation Summary

Tab. 12 summarizes the datasets and reward sources used in our experiments. We separate training corpora from held-out evaluation benchmarks when applicable. Mathematical tasks use rule-based answer extraction and verification, code

tasks use execution-based checking, and telecom tasks use model-based evaluation due to the open-ended nature of some responses.

Table 12. Dataset and evaluation summary.

Dataset	Domain	Use	Reward / Metric
Math-17k	Mathematical reasoning	RL training and evaluation	Rule-based final-answer accuracy
AIME24 / AIME25	Mathematical reasoning	Held-out evaluation	Rule-based final-answer accuracy
MATH500	Mathematical reasoning	Standard held-out evaluation	Rule-based final-answer accuracy
OpenR1-Code	Code generation	Auxiliary evaluation	Execution-based correctness
Datacom / Wireless	Telecom QA	Auxiliary evaluation	Model-based answer evaluation

C.2. Descriptive Statistical Summary

The main table reports mean and standard deviation over 5 independent runs. Tab. 13 provides a descriptive summary of the mean improvement and the standard error of the difference under an unpaired approximation. This table is intended as a diagnostic summary rather than a substitute for a paired significance test, since exact per-seed paired scores are not reported.

Table 13. Descriptive statistical summary for GRPO vs. GRPO+RLPT. The standard error of the difference uses an unpaired approximation with 5 runs per method.

Benchmark	Mean Improvement	Std. Error of Difference	Approx. t
Math-17k	$+2.00$	0.67	3.00
AIME24	$+1.66$	1.25	1.33
AIME25	$+2.20$	0.76	2.89
MATH500	$+3.48$	0.89	3.92

C.3. Cross-Ablation Interpretation

The support-alignment ablation in Sec. 6.3 is designed to isolate three effects: rollout truncation, optimization masking, and their alignment. Tab. 14 gives the intended interpretation of each row. The key observation is that neither rollout truncation alone nor optimization masking alone explains the full gain. The best result is obtained when the same behavior-policy support is used in both phases.

Table 14. Interpretation of the support-alignment ablation.

Variant	What It Tests
Top-K rollout, no optimization mask	Whether gains come merely from better rollout samples under Top-K decoding.
Top-p rollout, no optimization mask	Whether a common adaptive decoding heuristic already captures the effect.
Full rollout, Top-K optimization mask	Whether optimization masking works without matching the data-collection support.
Top-K rollout, same stored Top-K mask	Whether aligning rollout and optimization supports improves the RL update.

C.4. Additional Diagnostics

Explicit selector policy. As the core idea of this work is to define a compact optimization support, a natural alternative is to train an explicit, lightweight selector policy to filter tokens, rather than using the behavior policy's own distribution. We implement this baseline by training a multilayer perceptron selector that takes the context embedding and candidate token embeddings as input to output the index for token selection. We pre-train this selector on ALPACA (Taori et al., 2023), a general QA dataset, to learn general token relevance.

We compare this explicit selector method with RLPT on Math-17k, using GRPO as the base optimization algorithm. The results are shown in Fig. 6. We observe that the Explicit Selector method stops improving at a significantly lower score than RLPT. One likely reason is that the external selector is initialized from scratch and is not naturally calibrated to the base model's next-token distribution. In contrast, RLPT defines the support from the same behavior policy that generates the trajectory, preserving support consistency without an auxiliary selector.

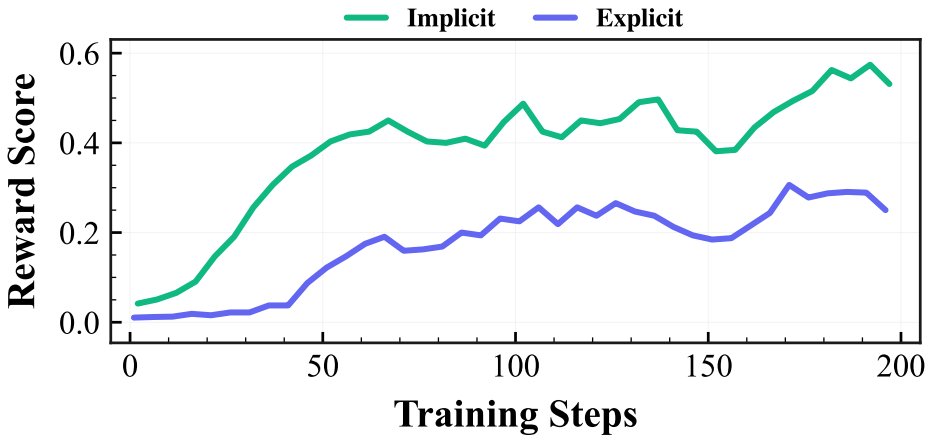

Figure 6. Comparison of RLPT training with explicit or implicit token selector.

Promising-set size. The hyperparameter K controls the trade-off between the expressiveness of the action space and the degree of support restriction. We evaluate $K \in \{4, 8, 16, 32\}$ on Math-17k, as shown in Fig. 7. A compact set of $K = 4$ yields the best asymptotic performance in our setup. Increasing K to 32 moves performance closer to the baseline, suggesting that overly large supports can reintroduce tail-token noise. Conversely, overly small supports may risk excluding useful continuations. In our mathematical reasoning setup, $K = 4 \sim 8$ provides the best empirical balance, but other domains may require larger or adaptive supports.

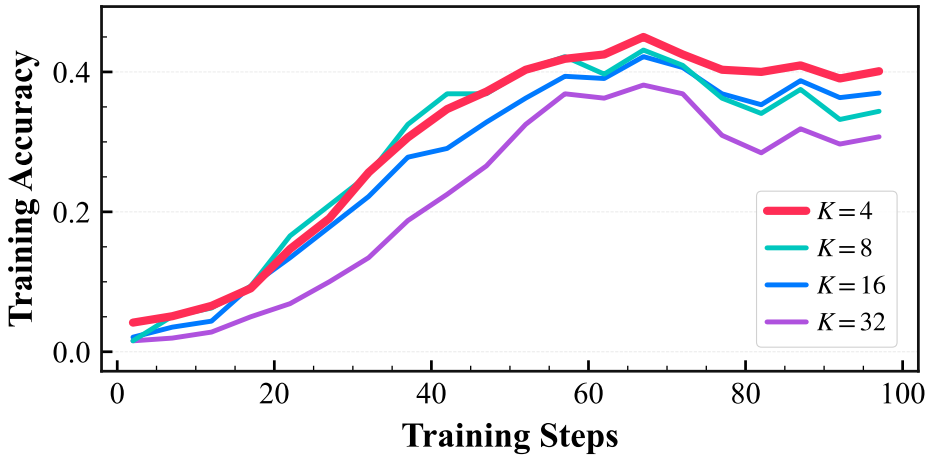

Figure 7. Ablation study on the size of promising set (K).

D. Omitted Proof

D.1. Proof of Proposition 4.1

Proposition 4.1 (Tail-gradient Removal) *For a fixed mask $\mathcal{P}_t$ and bounded advantage A_t, the masked policy gradient has zero logit-gradient components for all tail tokens in $\mathcal{T} = \mathcal{V} \setminus \mathcal{P}_t$. Therefore, the variance associated with these tail-logit components is removed relative to a full-vocabulary softmax estimator.*

Proof. First, we calculate the variance of the gradient component for a single token i. Since $a_t \sim \pi(\cdot|s_t)$, the term $\mathbb{I}(a_t = i)$ is a Bernoulli variable with parameter $\pi_i = \pi(i|s_t)$. The variance of $\hat{g}_i$ is:

$$\mathrm{Var}(\hat{g}_i) = \mathrm{Var}\left((\mathbb{I}(a_t = i) - \pi_i)A_t\right) = \pi_i(1 - \pi_i)A_t^2. \tag{7}$$

For the standard optimization over the full vocabulary $\mathcal{V}$, the tail contribution to the logit-gradient variance is:

$$\mathbb{V}_{\mathcal{T}}(\hat{g}_{\text{full}}) = \sum_{k \in \mathcal{T}_t} \pi_k(1 - \pi_k)A_t^2. \tag{8}$$

In RLPT, the policy is renormalized to $\tilde{\pi}$ over $\mathcal{P}_t$, and the logits for tail tokens $k \in \mathcal{T}_t$ are masked. Thus, their gradients are deterministically zero:

$$\hat{g}_{\text{mask},k} = 0, \quad \mathrm{Var}(\hat{g}_{\text{mask},k}) = 0, \quad \forall k \in \mathcal{T}_t. \tag{9}$$

Therefore, the tail-component variance removed by masking is:

$$\Delta\mathbb{V}_{\mathcal{T}} = \sum_{k \in \mathcal{T}_t} \pi_k(1 - \pi_k)A_t^2. \tag{10}$$

This statement concerns the tail-logit components only. The total variance over the remaining coordinates also depends on how probability mass is renormalized within $\mathcal{P}_t$, which is why we treat the result as a diagnostic explanation rather than a complete finite-sample convergence guarantee. □

E. More Details about Experiment Settings

E.1. Examples of the Datasets in Our Experiments

Tab. E.1 presents the examples from the datasets used for training and evaluation in our experiments.

E.2. Prompts for LLMs

- **Prompt for Math**: Let's think step by step. Output the final answer within `\boxed{}`, e.g., `\boxed{5}`.
- **Prompt for Code**: Solve the following coding problem using the programming language python: `{problems}`
- **Prompt for Domain**: For the following multiple-choice question, there is only one correct answer. Please analyze the question and the options, and place the correct option ID within `\boxed{}`, e.g., `\boxed{A}`.

E.3. Hyperparameters

The hyper-parameters for implementing RLPT and experiments are presented in Tab. 17. When implementing baseline methods, we use the same hyper-parameters as RLPT.

Dataset	Example Question	Target Answer
Math17k	"In triangle ABC, $\sin\angle A = \frac{4}{5}$ and $\angle A < 90^\circ$. Let D be a point outside triangle ABC such that $\angle BAD = \angle DAC$ and $\angle BDC = 90^\circ$. Suppose that $AD = 1$ and that $\frac{BD}{CD} = \frac{3}{2}$. If $AB + AC$ can be expressed in the form $\frac{a\sqrt{b}}{c}\ldots$"	"The answer is 34."
AIME24	"Every morning Aya goes for a 9-km-long walk and stops at a coffee shop. When she walks at s km/h, the walk takes 4 hours, including t minutes in the shop. When she walks $s+2$ km/h, it takes 2h 24m, including t minutes. If she walks at $s+\frac{1}{2}$ km/h, find the total minutes ..."	"The answer is 204."
AIME25	"In $\triangle ABC$, $D, E \in \overline{AB}$ and $F, G \in \overline{AC}$. Given $AD = 4, DE = 16, EB = 8$ and $AF = 13, FG = 52, GC = 26$. Let M be the reflection of D through F, and N be the reflection of G through E. If Area($DEGF$) = 288, find the area of heptagon $AFNBCEM$."	"The answer is 588."
OpenR1-Code	"Polycarp has n different binary words. He wants to reverse minimal number of words so that the final set can be arranged in a game sequence where each word starts with the previous word's last character ..."	"`if zo > oz: (zo-oz)//2 ...else: (oz-zo)//2 ...`"

Table 15. Examples from the mathematical reasoning datasets.

Domain	Example question	Target answer
Wireless-training	"What should I do if the DSP GTP-PATH command output shows that the GTP path is in the DEETECT state when the alarm is generated?"	"...1.First, check whether the peer GSN address specified in the alarm information is valid...;2.Next, execute the PING command to check if the link is normal...;3.Confirm whether the peer GSN can respond to ECHO messages..."
Wireless-testing	"During the deployment and operation of the LMT, the LMT is forcibly connected in HTTPS or WSS mode to ensure secure connection. In this mode, digital certificates are required for authentication. In addition, ...In such a deployment scenario, if the MAE of a carrier is set to HTTP login mode and the LMT is set to forcible HTTPS connection mode, what will happen? A. ...B. If the LMT connection mode is set to Force HTTPS, MAE proxy login fails to access the LMT due to protocol mismatch. The connection cannot be established even if the OM channel is normal. C. ...D. ..."	"The answer is X."
Datacom-training	"When configuring the Segment VXLAN feature, how can you enable EVPN as the VXLAN control plane on Transit Leaf1 and Transit Leaf2, and configure BGP EVPN peer relationships?"	"...1.Enter the BGP view or BGP multi-instance view...;2.Enter the BGP-EVPN address family view...;3.Configure the split group for BGP EVPN peers (groups)...;4.Enable the function to mark routes received from BGP EVPN peers as re-originated..."
Datacom-testing	"...The device supports creating subinterfaces on Layer 2 Ethernet and Layer 2 Eth-Trunk interfaces for VLAN termination to achieve inter-VLAN forwarding. However, the USG9500 series devices do not support creating subinterfaces on these two types of interfaces."	"The answer is error"

Table 16. Examples from the datasets used in our experiments.

Table 17. Hyper-parameters for training RLPT and baselines.

Hyper-parameters	**Value**
Batch size	8
Learning rate	10^{-6}
Learning rate decay style	$cosine$
Train iteration	200
Sequence length	4096
RL γ	1
RL λ	0.95
Mini batch size	4
RL clip ratio	0.2
Promising set size	4
Entropy coefficient	0.0
KL coefficient	0.01
Rollout count	8