Small RL Controller, Large Language Model: RL-Guided Adaptive Sampling for Test-Time Scaling

Runpeng Dai[1] Tong Zheng[2] Rui Liu[2] Chengsong Huang[3] Hongtu Zhu[1†]

[1]University of North Carolina at Chapel Hill [2]University of Maryland, College Park
[3]Washington University in St. Louis
{runpeng, htzhu}@unc.edu

Code: https://github.com/RunpengDai/RL-Guided-Adaptive-Sampling

Abstract

Test-time scaling improves the reasoning performance of large language models but incurs substantial cost in both total computation and latency. Existing adaptive sampling methods partially mitigate this issue by dynamically deciding when to stop sampling, yet they typically rely on heuristic rules or rely on distribution assumptions. In this work, we formulate adaptive sampling as a Markov decision process (MDP). We train a lightweight sampling controller with reinforcement learning (RL) to jointly balance answer correctness, latency, and computation cost. At each round, the controller decides to stop sampling or to acquire additional samples. Our method is lightweight which only relies on statistics of final answers, and can be trained and deployed on CPU. We further show that the resulting framework admits an interpretation as the Lagrangian relaxation of a constrained optimization problem with explicit budget constraints. Experiments against strong baselines such as ASC and ESC show that our method achieves improved trade-offs among answer correctness, sampling rounds, and total samples required.

1 Introduction

Test-time scaling (Snell et al., 2024; Zhang et al., 2025b) has emerged as an effective way to improve the reasoning performance of large language models (LLMs) without additional training. Methods such as self-consistency (Wang et al., 2022), tree-of-thoughts (Yao et al., 2023), and Best-of-N sampling (Nakano et al., 2021; Huang et al., 2025a) improve final answer quality by allocating more inference-time computation. However, these gains come with a clear downside: increased inference cost. As a result, effectively allocating inference-time computation to balance cost and performance is a critical challenge.

A growing line of work seeks to reduce the cost of test-time scaling through adaptive sampling. Early methods, such as Adaptive Self-Consistency (ASC) (Aggarwal et al., 2023), utilize the posterior answer distribution to determine whether additional samples are needed. Subsequent works build on this by incorporating semantic signals (Wan et al., 2025) or altering the prior distribution (Komiyama et al., 2026). Meanwhile, Early-Stopping Self-Consistency (ESC) (Li et al., 2024) reduces latency by shifting to a parallel execution strategy. A concurrent line of research attempts to alter the reasoning process itself. At inference time, some methods trigger early stopping via signals such as confidence (Fu et al., 2025), probing (Mao et al., 2025; Zheng et al., 2026a), or convergence dynamics (Liu and Wang, 2025; Zhang et al., 2025a). Alternatively, other works aim to increase reasoning efficiency during training time, leveraging either supervised fine-tuning (Xia et al., 2025; Munkhbat et al., 2025) or reinforcement learning (Aggarwal and Welleck, 2025).

While existing approaches are effective, they often suffer from several key limitations: (i) Many rely heavily on human-designed heuristics or distributional assumptions, rather than explicitly deriving an optimal policy to navigate the performance-cost trade-off. (ii) Several methods require auxiliary signals that are often unavailable under specific scenarios, such as internal model confidence, hidden states, or question difficulty. (iii) Some techniques are highly invasive, disrupting the model's natural reasoning process or even requiring additional training of the underlying LLM sampler. Such modifications incur significant operational overhead and are often incompatible with standard inference pipelines. These limitations motivate the need for *a lightweight but principled method for efficient test-time scaling*. Furthermore, to align with practical use cases, the method should flexibly account for multiple objectives, such as latency and computational cost.

In this work, we present a framework that cir-

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 37985–37999
October 24-29, 2026 ©2026 Association for Computational Linguistics

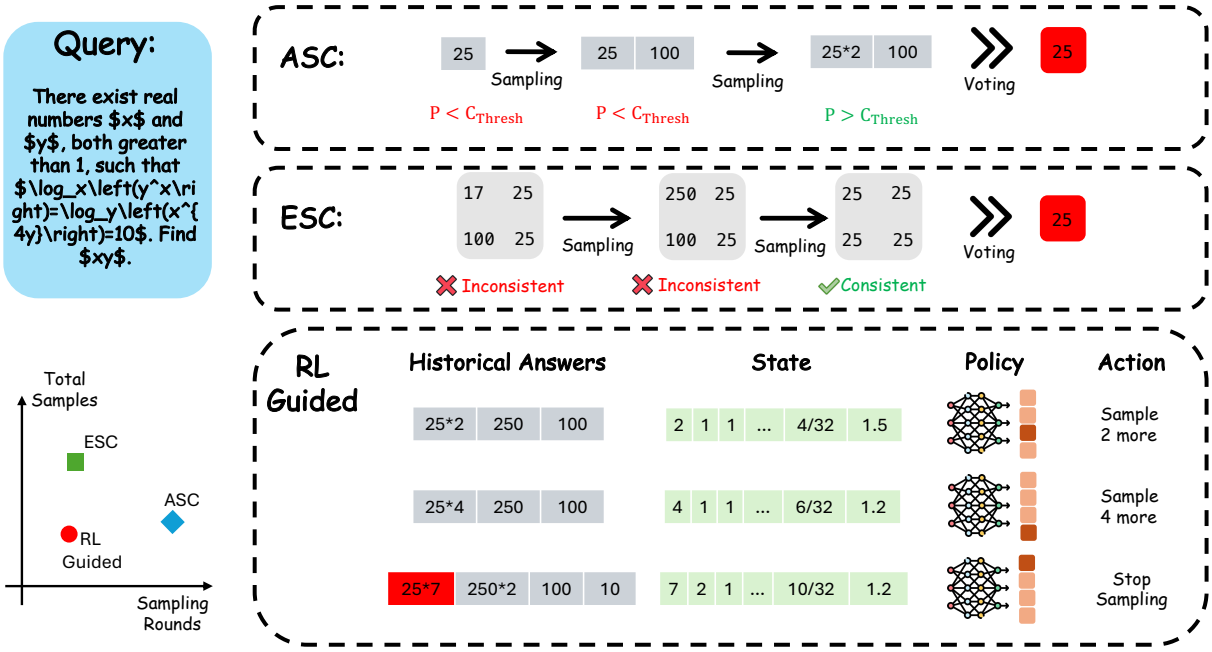

Figure 1: **Overview of the RL-Guided Sampling framework.** Top two blocks illustrates mechanism of two adaptive sampling baselines. ASC sequentially samples one response at a time and stops when the posterior probability exceeds a predefined threshold. ESC samples in fixed batches and stops only when intra-batch consistency is achieved. Different from those approaches, RL-Guided Sampling guides the sampling process via a lightweight policy network. At each round, the framework constructs a state based on statistics from the observed answer pool. Given this state, the policy network directs the language model to either sample a specific number of additional responses in parallel or halt generation. As summarized in the bottom-left plot, RL-Guided Sampling outperforms baseline methods by requiring fewer total samples and sampling rounds."

cumvents these limitations by formulating adaptive sampling as a Markov decision process (MDP). Rather than relying on predefined rules, we train a four-layer MLP controller via RL to learn a policy that optimizes the performance-cost trade-off. Crucially, our method relies purely on statistics derived from the sampled answer set, requiring only the final generated answers. This means it demands no auxiliary features, such as model confidence, and requires no intervention in the reasoning process of the LLMs. In the environment, we jointly consider multiple objectives. We use the correctness of the final answer as a positive reward, while treating inference costs (additional samples and sampling rounds) as penalties. Given an observed answer set, the controller dynamically decides whether to sample a specific number of additional responses or to stop sampling, as illustrated in Figure 1.

Practically, RL-Guided Sampling is extremely lightweight. The controller itself is only a small four-layer MLP, which can be trained and deployed efficiently on CPUs. Theoretically, our formulation intuitively maps to a constrained optimization problem: maximizing answer accuracy under strict latency and computation budgets. The weighted objective of our RL controller naturally arises as the Lagrangian relaxation of this problem, providing a principled mechanism to trade off performance and cost.

We validate RL-Guided Sampling across three benchmarks and multiple language-model samplers. Our experiments show that the proposed controller consistently improves the accuracy–efficiency trade-off over strong adaptive sampling baselines. Specifically, as shown in Table 1, RL-Guided Sampling reduces sampling rounds and total samples by 3 times and 30%, respectively, compared to ASC (Aggarwal et al., 2023), and by 10% and 35% compared to ESC (Li et al., 2024). This improvement is consistent across different trade-off levels (Figure 2). Furthermore, the learned policy generalizes well: controllers trained using samples from one dataset or model can be transferred to different benchmarks and even different samplers with minimal performance degradation (Section 4.5).

2 Method

2.1 Problem Setup and MDP Formulation

We formulate the adaptive sampling problem as a finite-horizon Markov decision process (MDP). At each round, the controller observes the statistical features of current pool of sampled answers, and then decides whether to stop or to acquire some additional samples. The reward is designed to reflect three factors: the quality of the final aggregated answer, the latency cost induced by additional sampling rounds, and the computation cost induced by generating more candidates.

Consider an input query x. Let N be the maximum number of samples allowed. At any given sampling round t, let $\mathcal{D}_t = \{y_1, \ldots, y_{n_t}\}$ denote the observed answer set, where each $y_i \in \mathcal{Y}$ is the final answer extracted from the i-th response, e.g., the content inside `\boxed{}`. Here, n_t denotes the total number of candidates collected up to round t.

State. The state s_t summarizes the evidence currently available to the controller, together with the inference cost already incurred. Specifically, the state consists of three components: the counts of the most frequent answer classes, the total number of sampled candidates and some light statistics. Let

$$\mathcal{V}_t = \{v_1(t), v_2(t), \ldots, v_K(t)\}$$

denote the sorted counts of the top-K most frequent answers in $\mathcal{D}_t$. We then represent the state using $\mathcal{V}_t$, the total number of samples generated $|\mathcal{D}_t|$, and the entropy of $\mathcal{V}_t$, such that $s_t = \{\mathcal{V}_t, |n_t|, \mathrm{Ent}(\mathcal{V}_t)\}$.

Action and transition. At each round, the controller selects an action from

$$\mathcal{A} = \{0, k_1, k_2, \ldots, k_L\}.$$

The action $a_t = 0$ means stopping the sampling process and returning the current majority-vote answer. An action $a_t = k_\ell > 0$ means generating k_ℓ additional candidate answers in parallel.

Following the generation of new samples, the observed answer set $\mathcal{D}_t$ is updated, and the state $\mathcal{V}_t$ is recomputed before advancing to the subsequent round. The episode terminates either when the controller chooses to stop (i.e., $a_t = 0$) or when the maximum sampling budget is reached (i.e., $n_t + a_t \geqslant N$).

Reward Function The reward function is designed to balance the final answer's quality against the inference cost. We can explicitly divide this into two distinct components: the step-wise penalty and the terminal reward.

Step-wise Penalty. At each round t, if the controller chooses to continue sampling by selecting a non-zero action $a_t > 0$, it incurs an intermediate penalty:

$$r_t^{\text{step}} = -\lambda_{\text{lat}} - \lambda_{\text{comp}} a_t,$$

where $\lambda_{\text{lat}}, \lambda_{\text{comp}} \geqslant 0$. Essentially, this penalizes the model by λ_{lat} for taking a new sampling step and by λ_{comp} for each new sample generated.

Terminal Reward. The episode terminates either when the controller chooses to stop or when the maximum sampling budget is reached. Upon termination, the controller aggregates the current candidate pool $\mathcal{D}_t$ via majority voting to output a final prediction:

$$\hat{y}_t = \mathrm{MajorityVote}(\mathcal{D}_t).$$

The model then receives a terminal reward r_t^{final} based on the correctness of the prediction:

$$r_t^{\text{final}} = \begin{cases} 1, & \text{if } \hat{y}_t = y^\star, \\ -1, & \text{otherwise.} \end{cases}$$

Specifically, we define $y^\star$ as the majority-vote answer that would be obtained if sampling continued to the maximum budget N without early stopping. This target encourages the controller to halt generation as soon as the current answer distribution converges. Notably, this reward design is intentionally decoupled from question-specific signals, such as ground-truth labels. This maintains consistency with our state representation, which depends solely on the sampled answer pool. We further validate these reward design choices through an ablation study in Section 4.6.

By combining these two components, the reward r_t at any sampling step t can be formally expressed as follows:

$$r_t = \mathbb{I}(a_t > 0) r_t^{\text{step}} + \mathbb{I}(a_t = 0 \text{ or } n_t + a_t \geqslant N) r_t^{\text{final}}$$

Optimization Objective. Given the per-step reward r_t, our goal is to train the controller to maximize the expected cumulative reward. Let t_{stop} denote the actual step at which the episode terminates. The optimization objective for the policy π_θ can be concisely formulated as:

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{t_{\text{stop}}} r_t\right], \tag{1}$$

where the expectation is taken over the sequence of states and actions induced by π_θ.

The proposed environment is compatible with a broad class of RL algorithms. In this work, we adopt Proximal Policy Optimization (PPO) as our default training algorithm, as it provides a stable on-policy framework for optimizing stochastic policies. However, alternative approaches, such as value-based methods (e.g., DQN) or other policy-gradient algorithms, can be readily applied within the same MDP formulation.

Furthermore, existing sampling strategies, such as ASC and ESC, can be naturally interpreted within this framework as fixed, rule-based policies operating over the proposed MDP. In contrast, our PPO-based controller dynamically learns an optimal strategy by explicitly maximizing the expected cumulative reward $J(\pi_\theta)$.

2.2 Lagrangian View

The optimization objective in Eq. (1) admits a simple Lagrangian interpretation. Intuitively, adaptive sampling can be viewed as a resource-constrained optimization problem: the controller seeks to maximize final answer quality while satisfying constraints on expected latency and computation. The penalties on sampling rounds and generated candidates then arise naturally as dual variables associated with these resource constraints.

Proposition 1 (Lagrangian interpretation)**.** *Let $J_{\text{ans}}(\pi_\theta)$, $J_{\text{comp}}(\pi_\theta)$, and $J_{\text{lat}}(\pi_\theta)$ denote the expected accuracy, total number of samples, and number of sampling rounds under policy π_θ, respectively. Consider the budget-constrained adaptive sampling problem*

$$\begin{aligned} \max_{\pi} \quad & J_{\text{ans}}(\pi_\theta) \\ \text{s.t.} \quad & J_{\text{comp}}(\pi_\theta) \leqslant C_{\text{comp}}, \\ & J_{\text{lat}}(\pi_\theta) \leqslant C_{\text{lat}}. \end{aligned}$$

Optimizing the objective in Equation 1 is equivalent to optimizing the Lagrangian relaxation of this constrained problem with respect to π, where the non-negative penalty weights act as dual variables associated with the sample and round constraints.

We provide the formal statement and proof in Appendix C. Proposition 1 provides an interpretation of our reward design from a constrained optimization perspective. This connects RL-Guided Sampling to a broader literature on constrained and safe RL (Garcıa and Fernández, 2015; Altman, 2021), where policies are optimized under explicit cost or safety constraints. While we use fixed penalty in this work, this perspective suggests a promising future direction: directly optimizing policies under prescribed budgets with constrained RL methods.

2.3 Discussion: Generality of the Framework

The RL-guided sampling framework is not tied to a particular MDP design. In the default setting, we keep the formulation simple by using only lightweight statistics derived from the sampled final answers and adopting majority voting for aggregation. However, when richer signals are available, they can be naturally incorporated into the same MDP framework without altering its overall formulation.

For example, if confidence scores are available (Taubenfeld et al., 2025a), they can be used in two complementary ways. First, confidence can be incorporated into the state representation, allowing the controller to capture the model's confidence in different answer classes. Second, confidence can be incorporated into the aggregation rule itself, replacing standard majority voting with a confidence-weighted aggregation strategy. We empirically investigate these confidence-augmented variants in Section 4.4.

3 Experimental Settings

3.1 Dataset and Samplers

In this paper, we evaluate our method on questions from three challenging mathematical reasoning benchmarks: AIME24, AIME25 (MAA), and HMMT 2025 (Dekoninck et al., 2026). This benchmark selection aligns with prior work (Zheng et al., 2026a), which is intended to provide a balanced level of difficulty. To train the RL-guided sampling controller, we randomly sample a 200-question subset from the DAPO training set (Yu et al., 2025).

We further evaluate the scalability and generalizability of RL-guided sampling across a diverse set of LLM samplers. Specifically, we consider multiple variants of the Qwen-3 family (Yang et al., 2025) and the closed-source GPT-4.1-nano model (Achiam et al., 2023). These samplers span different model scales (0.6B, 1.7B, and 4B), model types (reasoning and instruct), and deployment settings (open-source and proprietary). This broad selection allows us to examine whether the benefits of RL-guided sampling transfer consistently from lightweight open-source models to more capable

proprietary systems.

3.2 Baseline Methods and Evaluation Metrics

To evaluate the effectiveness of RL-Guided Sampling, we compare it against representative test-time scaling baselines:

- **SC** (Self-Consistency (Wang et al., 2022)): A standard test-time scaling method that samples multiple independent reasoning trajectories in parallel and returns the majority-voted answer.
- **ASC** (Adaptive Self-Consistency (Aggarwal et al., 2023)): An adaptive sampling method that sequentially samples one response at a time, updates the posterior distribution, and stops once a predefined p-value threshold is reached.
- **ESC** (Early Stopping Consistency (Li et al., 2024)): A chunk-based approach that generates a fixed number of trajectories in parallel at each step and terminates early when the responses within a batch are consistent.

We report performance using five key metrics, grouped into three categories. First, **Accuracy** measures the percentage of correctly solved problems. Second, **Total Samples** and **Total Tokens** measure computational cost: the former is directly aligned with our environment design, while the latter reflects the actual token cost during deployment. Third, **Sampling Rounds** and **Sequential Tokens** measure latency: the former captures the number of adaptive sampling steps, while the latter measures the latency-critical sequential token length in real-time inference.

3.3 Environment and Training Setup

For state construction, we set $K = 5$, resulting in an 7-dimensional state space. The action space $\mathcal{A} = \{0, 1, 2, 4\}$ additional responses. The policy network is a four-layer MLP. Full implementation details of the RL training procedure and environment setup are provided in Appendix A.

4 Results and Analysis

4.1 Main Results

The main results are reported in Table 1 and Table 6 in the Appendix. The former reports performance in terms of sampling rounds and total samples, while the latter provides token-level metrics, including total tokens and sequential tokens.

Overall, RL-Guided Sampling consistently achieves a better accuracy–efficiency trade-off than strong baselines. The key observations are as follows:

- ASC effectively reduces total sample consumption but suffers from high latency due to its reliance on excessive sequential sampling rounds. In contrast, ESC evaluates on a coarse grid to reduce these rounds, but it compromises performance by either requiring more total samples or sacrificing accuracy.
- Compared to ASC, RL-Guided Sampling significantly alleviates the latency bottleneck by reducing average sampling rounds by 3–4$\times$ while preserving comparable accuracy. Furthermore, it achieves this efficiency while reducing the total number of samples by approximately 30%.
- Compared to ESC, RL-Guided Sampling avoids overly aggressive early stopping to achieve a better overall trade-off. It slightly reduces the number of sampling rounds by approximately 10%, while significantly lowering the total sample requirement by roughly 33% and achieving higher accuracy.
- Compared to SC, all adaptive methods (ASC, ESC, and RL-Guided Sampling) significantly reduce total sample count. Nevertheless, their sampling rounds inevitably exceed those of SC due to the sequential evaluations required for adaptive early stopping.

4.2 Scaling Curves

Figure 2 illustrates the test-time scaling behavior of RL-Guided Sampling under various hyperparameter settings, comparing it against SC, ASC, and ESC using Qwen3-4B-Instruct on the AIME 2024 and 2025 datasets. Each curve represents a parameter sweep, capturing different preferences along the accuracy–efficiency trade-off. Detailed experimental setups and additional scaling results for other models and datasets are provided in Appendix B.2.

Overall, these results align with the findings in Section 4.1, demonstrating that RL-Guided Sampling consistently achieves a favorable trade-off from both efficiency perspectives. In terms of total sample budget, RL-Guided slightly outperforms ASC and significantly dominates both ESC and SC across most operating points. Furthermore, when evaluated by sampling rounds, both RL-Guided

Model: InCoder-6B; train: MBPP Method	MBPP				HumanEval				Average			
	Agree.	Code Acc.	Samples	Tokens	Agree.	Code Acc.	Samples	Tokens	Agree.	Code Acc.	Samples	Tokens
ASC ($\tau = 0.95$)	99.9	26.5	26.9	1.7k	100.0	16.1	27.6	2.3k	99.9	21.3	27.2	2.0k
ESC ($w = 5$)	98.1	26.3	30.8	2.0k	98.3	16.1	30.9	2.5k	98.2	21.2	30.8	2.3k
RL-Guided ($\lambda_c = 0.0075$)	$99.2_{\pm 0.77}$	$26.4_{\pm 0.02}$	29.6	1.9k	$99.3_{\pm 0.71}$	$16.1_{\pm 0.01}$	29.8	2.4k	99.3	21.3	29.7	2.1k

Table 1: **Comparison of test-time scaling approaches across three benchmarks.** Acc. denotes accuracy, Rounds measures the number of adaptive sampling rounds, and # Samples counts the total number of sampled responses. In general, better methods achieve higher Acc. with fewer Rounds and # Samples. For the Self-Consistency baseline, we use 32 samples, denoted as SC@32. For ASC and ESC, we follow the default settings from the original papers, setting the ASC threshold to 0.95 and the ESC chunk size to 5. For RL-guided adapters, we report the mean performance over five random seeds, together with the standard deviation.

and ESC maintain strong performance while requiring significantly fewer rounds. Ultimately, these results validate the effectiveness and stability of RL-Guided Sampling across a broad spectrum of parameter configurations, demonstrating robust performance well beyond default settings.

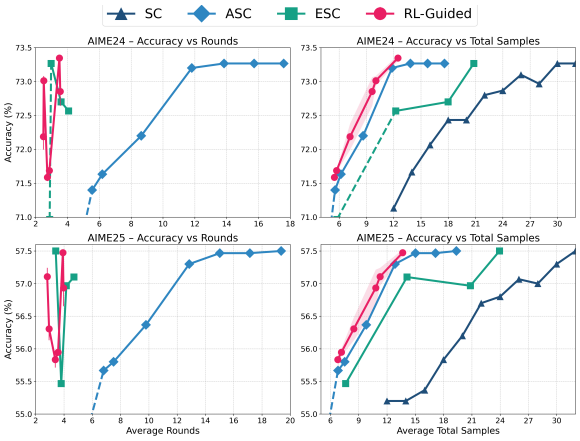

Figure 2: Accuracy scaling behavior across sampling budgets. Left: Accuracy vs. Sampling Rounds. Right: Accuracy vs. Total Samples. Results are generated with Qwen3-4B-Instruct on the AIME24 and AIME25 datasets. Compared to SC, ASC, and ESC, RL-Guided sampling consistently achieves superior accuracy under the same or fewer samples and rounds. Analogous scaling curves measured by token consumption are presented in Figure 5.

4.3 Explanatory Analysis

Compared to Self-Consistency (SC) with a fixed sample size, the primary advantage of adaptive sampling lies in its ability to dynamically allocate computational resources across queries. In this section, we investigate how RL-Guided Sampling distributes this computation. Specifically, we record the average total samples consumed per query and examine its correlation with two established query-level metrics, as illustrated in Figure 3.

Given a specific query, we define the following two metrics: **Answer Entropy** is the Shannon entropy of the categorical distribution over the final answers, treating each unique generated answer as a distinct category. **Answer Accuracy** denotes the empirical probability of generating a correct response, calculated over the total sampled responses.

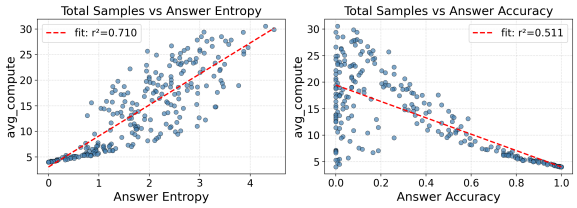

Figure 3: Correlation between the average total samples per query and Answer Entropy (left) alongside Answer Accuracy (right). Each data point represents a distinct query from the DAPO-subset, with responses generated by the Qwen3-0.6B model.

The results demonstrate that RL-Guided sampling learns an intuitive resource allocation strategy. Specifically, as illustrated in Figure 3, the average sample budget allocated by the policy exhibits a distinct positive correlation with Answer Entropy. This aligns with the intuition that queries generating highly diverse answers inherently require more samples to converge and form a reliable majority. Notably, the resource allocation does not strictly mirror the entropy score, suggesting that the learned policy captures the nuanced dynamics of the sampling trajectory rather than applying a rigid heuristic threshold. In contrast, the correlation with Answer Accuracy is comparatively weak. This aligns with our formulation, as the RL controller relies purely on the real-time statistics of the ongoing sampling process without incorporating query-level information.

Reward Signal	AIME24			AIME25			HMMT25			Avg.		
	Acc. ↑	Rounds ↓	#Samples ↓	Acc. ↑	Rounds ↓	#Samples ↓	Acc. ↑	Rounds ↓	#Samples ↓	Acc. ↑	Rounds ↓	#Samples ↓
Running Majority	$73.0_{\pm 0.07}$	2.5	10.0	$57.1_{\pm 0.14}$	2.8	11.0	$33.6_{\pm 0.03}$	3.0	11.8	54.6	2.8	10.9
Full Majority	$73.0_{\pm 0.11}$	2.6	11.0	$57.1_{\pm 0.19}$	2.9	12.0	$33.6_{\pm 0.04}$	3.0	12.2	54.6	2.8 (+2.4%)	11.7 (+7.3%)
Real Label	$71.9_{\pm 0.27}$	3.7	15.0	$55.7_{\pm 0.39}$	4.3	18.0	$33.6_{\pm 0.12}$	4.3	18.0	53.7	4.1 (+48.2%)	17.0 (+55.9%)

Table 2: Ablation study evaluating the impact of different reward signals across three reasoning benchmarks using Qwen3-4B-Instruct. We compare our default *Running Majority* target against a *Full Majority* target and the ground-truth *Real Label*. The *Running Majority* configuration achieves the best overall performance. Notably, utilizing the *Real Label* as the reward signal results in a substantial degradation in accuracy, accompanied by significant increase in both sampling rounds and total generated samples. All experimental settings are identical to those in Table 1.

4.4 Confidence-Augmented Sampling

We further investigate whether the proposed framework can benefit from confidence signals. Following CISC (Taubenfeld et al., 2025a), we use the model's $P(\text{True})$ confidence score and conduct the following two experiments:

- **RL-guided + CISC:** Replace majority voting with confidence-weighted aggregation at evaluation time, without changing the MDP formulation or the trained controller.
- **Confidence-aware RL-guided:** Augment each answer class in the state with its average confidence and replace majority voting in the MDP environment with confidence-weighted aggregation.

The results are summarized in Table 3. As shown, replacing majority voting with confidence-weighted aggregation substantially improves the accuracy of the original RL-guided controller without changing the environment. Further incorporating confidence into the MDP state yields an additional improvement. These results demonstrate that the proposed framework can naturally exploit richer signals when available: confidence can improve both how sampled answers are aggregated and how the controller decides whether additional sampling is necessary.

4.5 Generalization Analysis

Having previously validated that RL-Guided Sampling generalizes across different datasets using the same model in section 4.1 and 4.2, we now evaluate a more challenging setting: generalizing across *both* models and datasets simultaneously. Specifically, Figure 4 illustrates the performance of a controller trained on the Qwen3-0.6B model when applied directly to guide inference for the closed-source GPT-4.1-nano model. As shown, the learned policy demonstrates strong robustness, maintaining highly competitive scaling behavior across various parameter settings despite the distribution shift.

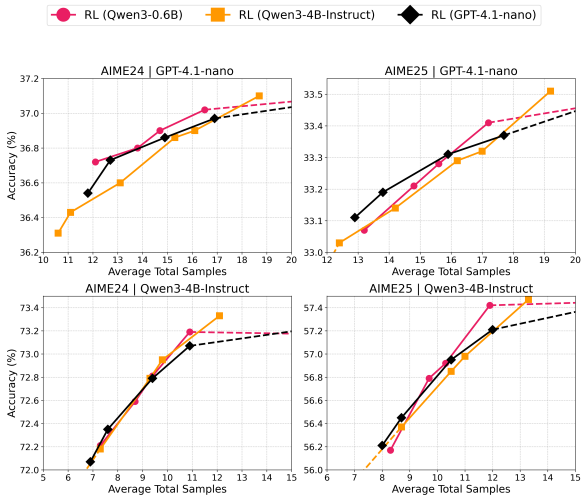

Figure 4: Accuracy scaling behavior of RL-Guided Sampling evaluated on GPT-4.1-nano (top) and Qwen3-4B-Instruct (bottom). Lines denote controllers trained on the DAPO-subset using responses generated by different models, demonstrating robust cross-model transferability.

This strong generalizability carries significant practical implications. It suggests that the controller successfully captures model-agnostic signals of sampling convergence rather than overfitting to a specific model's distribution. Consequently, practitioners can train the policy using existing data or inexpensive samples from a lightweight open-source model, and seamlessly apply it to reduce inference costs for new, stronger, or closed-source models where direct sampling is costly or infeasible.

4.6 Ablation Study

We investigate the impact of different reward signals by comparing three target label ($y^\star$) constructions for the terminal reward: Running Majority

Table 3: Confidence-augmented adaptive sampling results on AIME24 and AIME25. We report accuracy and the average number of samples used.

Method	AIME24		AIME25		Average	
	Acc. ↑	# Samples ↓	Acc. ↑	# Samples ↓	Acc. ↑	# Samples ↓
ASC + CISC	78.9	11.3	59.4	12.3	69.2	11.8
RL-guided + CISC	78.0	**9.4**	60.1	**10.7**	69.0	**10.0**
Confidence-aware RL-guided + CISC	**78.8**	10.1	**60.2**	11.4	**69.5**	10.8

(default; the majority vote if sampling continues to the $N = 32$ budget, Full Majority (the majority vote from the full answer pool of 128 samples), and Real Label (the ground-truth answer). The results are summarized in Table 2.

As shown, the Running Majority target yields optimal accuracy-efficiency trade-off. Conversely, utilizing the Real Label significantly degrades accuracy and increases sample cost. This underperformance relates to our state representation. Because the state excludes the semantic information of the query, the controller must base its decisions on the statistical distribution of the sampled answer pool. Consequently, the controller lacks the context to verify factual accuracy. Using the Real Label therefore introduces indistinguishable noise into the optimization process. Furthermore, incorporating such problem-dependent signals prevents the policy from learning universal stopping criteria, ultimately degrading its generalization performance.

Furthermore, Running Majority slightly outperforms Full Majority in sampling efficiency. We hypothesize that the actual samples limit ($N = 32$) provides a more attainable positive reward signal compared to forcing the policy to predict the consensus of a much larger sample pool.

5 Related works

5.1 Efficient Parallel Reasoning

A recent line of work seeks to reduce the cost of fixed-budget parallel sampling through dynamic resource allocation. Aggarwal et al. (2023) and Li et al. (2024) terminate sampling once a consensus criterion is reached, and Wang et al. (2025a) further adapt the sample budget to query difficulty. A complementary direction weights reasoning paths by confidence to recover high-quality answers from fewer samples (Huang et al., 2025b; Taubenfeld et al., 2025b; Fu et al., 2025). These methods, however, largely rely on sequential sampling, which undermines the hardware advantage of parallel decoding. Finer-grained schemes such as Dynamic Self-Consistency (Wan et al., 2025), Self-Truncation (Wang et al., 2025b), DeepPrune (Tu et al., 2025), Step (Liang et al., 2026), Slim-SC (Hong et al., 2025), Parallel-Probe (Zheng et al., 2026a) instead prune unpromising trajectories during generation to avoid wasted compute on incorrect paths. More recently, AutoTTS (Zheng et al., 2026b) explores letting LLMs themselves discover better test-time scaling strategies. Yet these methods rely on intrinsic signals (e.g., logits) or mid-generation rollouts to decide when to prune, complicating deployment. We instead cast adaptive batched test-time scaling as an MDP and search for a lightweight controller that jointly optimizes accuracy, latency, and cost, yielding better Pareto trade-offs.

5.2 Test-Time Scaling

Improving the efficiency of complex reasoning has increasingly been framed as a question of how to allocate test-time computation (Snell et al., 2024; Chen et al., 2025b; Wang et al., 2026a; Huang et al., 2026). A prominent instantiation is tree search, which aggregates diverse reasoning paths and uses sparse activation to keep the search tractable (Bi et al., 2024; Lample et al., 2022; Koh et al., 2024; Zheng et al., 2025). Step-wise verifiers further tighten this search by pruning unproductive branches on the fly (Wang et al., 2022; Li et al., 2022; Lightman et al., 2023). Orthogonal to search itself, gains have been reported from diversifying query formulations (Huang et al., 2024) and from iterative refinement loops that bootstrap the model's self-correction on harder problems (Chen et al., 2025a; Welleck et al., 2022; Madaan et al., 2023; Aggarwal et al., 2024; Wang et al., 2026b).

5.3 Multi-Objective and Constrained Reinforcement Learning

Multi-Objective Reinforcement Learning (MORL) addresses sequential decision-making problems governed by conflicting goals, such as maximizing diagnostic yield while minimizing invasive tests

in healthcare (Qiu et al., 2026). By optimizing across multiple reward signals, MORL seeks to discover a set of Pareto-optimal policies that capture different trade-offs (Roijers et al., 2013; Hayes et al., 2022). A foundational approach in MORL is linear scalarization, which aggregates the multi-dimensional objective into a single scalar reward using pre-defined preference weights (Parisi et al., 2014; Mossalam et al., 2016). Alternatively, conditioned approaches incorporate the preference vector directly into the state space, aiming to learn a single unified policy $\pi(a|s, w)$ that generalizes across the entire Pareto front (Abels et al., 2019; Yang et al., 2019; Navon et al., 2020). Given its algorithmic simplicity and robust empirical performance, our proposed method relies on the linear scalarization paradigm.

Constrained Reinforcement Learning (CRL) focuses on maximizing a primary objective while satisfying several limits, such as safety boundaries or resource budgets (Garcıa and Fernández, 2015; Altman, 2021). Through Lagrangian relaxation, this constrained formulation is fundamentally connected to MORL. Mainstream CRL methods employ dynamic dual updates or bounded optimizations to strictly enforce these constraints (e.g., Constrained Policy Optimization (Achiam et al., 2017) and primal-dual approaches (Tessler et al., 2018)).

6 Conclusion

This paper introduces RL-Guided Sampling, a principled and lightweight framework for efficient test-time scaling. By formulating adaptive sampling as a Markov decision process, we train an RL controller to explicitly optimize the trade-off among answer correctness, computational cost, and latency. Unlike existing approaches, RL-Guided Sampling is non-invasive and relies purely on the statistics of the generated answers, eliminating the need for auxiliary signals. Empirically, our CPU-friendly controller consistently outperforms strong baselines such as ASC and ESC, significantly reducing both total samples and sampling rounds. Furthermore, the learned policy exhibits strong transferability across different datasets and samplers.

Limitations

This work takes an initial step toward formulating adaptive LLM sampling as a reinforcement learning problem. While our lightweight controller already shows consistent improvements in the accuracy–latency–compute trade-off, the current formulation can be further refined.

In particular, our state representation intentionally relies on simple statistics of the sampled answers, leaving room to incorporate richer signals such as answer confidence, or the average length of answers. More importantly, on reward design, future work could directly include real-life costs, such as directly using time and money cost of the generating process as penalty. These extensions are complementary to our framework and may further improve its alignment with real-world deployment costs.

References

Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic weights in multi-objective deep reinforcement learning. In *International conference on machine learning*, pages 11–20. PMLR.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. Pmlr.

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and 1 others. 2023. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with llms. *arXiv preprint arXiv:2305.11860*.

Pranjal Aggarwal, Bryan Parno, and Sean Welleck. 2024. Alphaverus: Bootstrapping formally verified code generation through self-improving translation and treefinement.

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

Eitan Altman. 2021. *Constrained Markov decision processes*. Routledge.

Bin Bi and 1 others. 2024. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *ArXiv preprint*, abs/2412.09078.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym.

Jiefeng Chen, Jie Ren, Xinyun Chen, Chengrun Yang, Ruoxi Sun, and Sercan Ö Arık. 2025a. Sets: Leveraging self-verification and self-correction for improved test-time scaling. *ArXiv preprint*, abs/2501.19306.

Weizhe Chen, Sven Koenig, and Bistra Dilkina. 2025b. Iterative deepening sampling as efficient test-time scaling. *arXiv preprint arXiv:2502.05449*.

Jasper Dekoninck, Nikola Jovanović, Tim Gehrunger, Kári Rögnvaldsson, Ivo Petrov, Chenhao Sun, and Martin Vechev. 2026. Beyond benchmarks: Matharena as an evaluation platform for mathematics with llms.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. 2025. Deep think with confidence. *ArXiv*, abs/2508.15260.

Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480.

Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, and 1 others. 2022. A practical guide to multi-objective reinforcement learning and planning: Cf hayes et al. *Autonomous Agents and Multi-Agent Systems*, 36(1):26.

Colin Hong, Xu Guo, Anand Chaanan Singh, Esha Choukse, and Dmitrii Ustiugov. 2025. Slim-sc: Thought pruning for efficient scaling with self-consistency. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 34488–34505.

Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. 2025a. Is best-of-n the best of them? coverage, scaling, and optimality in inference-time alignment. *arXiv preprint arXiv:2503.21878*.

Chengsong Huang, Langlin Huang, and Jiaxin Huang. 2024. Divide, reweight, and conquer: A logit arithmetic approach for in-context learning. *ArXiv preprint*, abs/2410.10074.

Chengsong Huang, Langlin Huang, Jixuan Leng, Jiacheng Liu, and Jiaxin Huang. 2025b. Efficient test-time scaling via self-calibration. *arXiv preprint arXiv:2503.00031*.

Chengsong Huang, Tong Zheng, Langlin Huang, Jinyuan Li, Haolin Liu, and Jiaxin Huang. 2026. Relayllm: Efficient reasoning via collaborative decoding. *arXiv preprint arXiv:2601.05167*.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. 2024. Tree search for language model agents.

Junpei Komiyama, Daisuke Oba, and Masafumi Oyamada. 2026. Best-of-infinity: Asymptotic performance of test-time llm ensembling. In *The Fourteenth International Conference on Learning Representations*.

Guillaume Lample, Marie-Anne Lachaux, Thibaut Lavril, Xavier Martinet, Amaury Hayat, Gabriel Ebner, Aurélien Rodriguez, and Timothée Lacroix. 2022. Hypertree proof search for neural theorem proving.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022. Making large language models better reasoners with step-aware verifier.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. 2024. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. *arXiv preprint arXiv:2401.10480*.

Zhixiang Liang, Beichen Huang, Zheng Wang, and Minjia Zhang. 2026. Hidden states as early signals: Step-level trace evaluation and pruning for efficient test-time scaling. *arXiv preprint arXiv:2601.09093*.

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

Xin Liu and Lu Wang. 2025. Answer convergence as a signal for early stopping in reasoning. *arXiv preprint arXiv:2506.02536*.

MAA. n.d. American invitational mathematics examination (AIME). Mathematics Competition Series.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback.

Minjia Mao, Bowen Yin, Yu Zhu, and Xiao Fang. 2025. Early stopping chain-of-thoughts in large language models. *ArXiv*, abs/2509.14004.

Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. 2016. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*.

Tergel Munkhbat, Namgyu Ho, Seo Hyun Kim, Yongjin Yang, Yujin Kim, and Se-Young Yun. 2025. Self-training elicits concise reasoning in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 25127–25152.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, and 1 others. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. 2020. Learning the pareto front with hypernetworks. *arXiv preprint arXiv:2010.04104*.

Simone Parisi, Matteo Pirotta, Nicola Smacchia, Luca Bascetta, and Marcello Restelli. 2014. Policy gradient approaches for multi-objective sequential decision making. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2323–2330. IEEE.

Jiaming Qiu, Ying-Qi Zhao, John Wei, Arul M Chinnaiyan, Jeffrey Tosoian, and Yingye Zheng. 2026. Optimizing sequential decision rules for prostate cancer biopsy management: A multi-objective statistical framework. *Journal of the American Statistical Association*, (just-accepted):1–24.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Amir Taubenfeld, Tom Sheffer, Eran Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and Gal Yona. 2025a. Confidence improves self-consistency in llms. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 20090–20111.

Amir Taubenfeld, Tom Sheffer, Eran. O. Ofek, Amir Feder, Ariel Goldstein, Zorik Gekhman, and G. Yona. 2025b. Confidence improves self-consistency in llms. In *Annual Meeting of the Association for Computational Linguistics*.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. 2018. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*.

Shangqing Tu, Yaxuan Li, Yushi Bai, Lei Hou, and Juanzi Li. 2025. Deepprune: Parallel scaling without inter-trace redundancy. *arXiv preprint arXiv:2510.08483*.

Guangya Wan, Yuqi Wu, Jie Chen, and Sheng Li. 2025. Reasoning aware self-consistency: Leveraging reasoning paths for efficient llm sampling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3613–3635.

Xinglin Wang, Shaoxiong Feng, Yiwei Li, Peiwen Yuan, Yueqi Zhang, Chuyi Tan, Boyuan Pan, Yao Hu, and Kan Li. 2025a. Make every penny count: Difficulty-adaptive self-consistency for cost-efficient reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 6904–6917.

Xinglin Wang, Zishen Liu, Shaoxiong Feng, Peiwen Yuan, Yiwei Li, Jiayi Shi, Yueqi Zhang, Chuyi Tan, Ji Zhang, Boyuan Pan, and 1 others. 2026a. On time, within budget: Constraint-driven online resource allocation for agentic workflows. *arXiv preprint arXiv:2605.06110*.

Xinglin Wang, Jiayi Shi, Shaoxiong Feng, Peiwen Yuan, Yiwei Li, Yueqi Zhang, Chuyi Tan, Ji Zhang, Boyuan Pan, Yao Hu, and 1 others. 2026b. Do not waste your rollouts: Recycling search experience for efficient test-time scaling. *arXiv preprint arXiv:2601.21684*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Yiming Wang, Pei Zhang, Siyuan Huang, Baosong Yang, Zhuosheng Zhang, Fei Huang, and Rui Wang. 2025b. Sampling-efficient test-time scaling: Self-estimating the best-of-n sampling in early decoding. *arXiv preprint arXiv:2503.01422*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct.

Heming Xia, Chak Tou Leong, Wenjie Wang, Yongqi Li, and Wenjie Li. 2025. Tokenskip: Controllable chain-of-thought compression in llms. *arXiv preprint arXiv:2502.12067*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. 2019. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. *Advances in neural information processing systems*, 32.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Junyu Zhang, Runpei Dong, Han Wang, Xuying Ning, Haoran Geng, Peihao Li, Xialin He, Yutong Bai, Jitendra Malik, Saurabh Gupta, and 1 others. 2025a. Alphaone: Reasoning models thinking slow and fast at test time. *arXiv preprint arXiv:2505.24863*.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, and 1 others. 2025b. A survey on test-time scaling in large language models: What, how, where, and how well? *arXiv preprint arXiv:2503.24235*.

Tong Zheng, Chengsong Huang, Runpeng Dai, Yun He, Rui Liu, Xin Ni, Huiwen Bao, Kaishen Wang, Hongtu Zhu, Jiaxin Huang, and 1 others. 2026a. Parallel-probe: Towards efficient parallel thinking via 2d probing. *arXiv preprint arXiv:2602.03845*.

Tong Zheng, Haolin Liu, Chengsong Huang, Huiwen Bao, Sheng Zhang, Rui Liu, Runpeng Dai, Ruibo Chen, Chenxi Liu, Tianyi Xiong, and 1 others. 2026b. Llms improving llms: Agentic discovery for test-time scaling. *arXiv preprint arXiv:2605.08083*.

Tong Zheng, Hongming Zhang, Wenhao Yu, Xiaoyang Wang, Runpeng Dai, Rui Liu, Huiwen Bao, Chengsong Huang, Heng Huang, and Dong Yu. 2025. Parallel-r1: Towards parallel thinking via reinforcement learning. *arXiv preprint arXiv:2509.07980*.

A Implementing Details

For each question, we generate 128 candidate responses. The sampling configuration largely follows the recommended parameters for each model, as summarized in Table 4. During both training and evaluation, each trajectory is constructed by randomly sampling $N = 32$ responses from this candidate pool. We use the following prompt:

> {Query}
> Please reason step by step, and put your final answer within `\boxed{}`.

Model	Max Length	Temperature	Top-k	Top-p
Qwen3-0.6B-Thinking	32768	0.6	20	0.95
Qwen3-1.7B-Thinking	32768	0.6	20	0.95
Qwen3-4B-Instruct	32768	0.7	20	0.8
GPT-4.1-nano	32768	0.8	20	0.95

Table 4: Sampling configurations used to generate candidate responses for each model.

The RL adapter is a four layer neural network. We train the RL guided adapter with the PPO method implemented with the Stable-Baselines3 framework (Raffin et al., 2021) and the MDP environment is built following the OpenAI gym framework (Brockman et al., 2016). Detailed training configurations for both methods are provided in Table 5. The experiments are conducted on a linux platform with 64 INTEL(R) XEON(R) GOLD 6548Y+ CPU.

During training, we first randomly select questions from the DAPO subset. For each selected question, we construct a trajectory by randomly sampling a batch $N = 32$ responses from its offline response pool of 128 candidates, and use the majority-vote answer within this batch as the pseudo-label y^*. During evaluation, we iterate over all questions in each benchmark. For each question, we repeatedly sample 32 responses from the corresponding pool of 128 candidates and evaluate all methods under the same sampled response sets. We repeat this process over 100 random seeds and report the mean accuracy and computation cost.

Parameter	Value
learning_rate	1×10^{-5}
total_timesteps	1×10^{6}
gamma (Discount Factor)	1
gae_lambda	0.95
Hidden layers	[32, 64, 64, 32]

Table 5: Training hyperparameter of RL-Guided Sampling.

B Additional Experimental Results

B.1 Additional Details and Results for Main Results

For the results in Table 1, we use a default penalty setting of $\lambda_{\text{lat}} = 0$ and $\lambda_{\text{comp}} = 0.0075$ when training the RL-guided controller for all models. For each sampler, the controller is trained on the DAPO subset using responses generated by that same sampler. To account for randomness in RL training, we train five controllers with different random seeds for each model. We report the mean sampling rounds, total number of samples, and accuracy across these runs, with the standard deviation reported for accuracy.

In addition to Table 1, we report the performance of RL-Guided Sampling and competing baselines using token-level cost metrics in Table 6. Specifically, we measure computation cost by Total Tokens and latency by Sequential Tokens. The results are consistent with those in Section 4.1, further confirming the accuracy–efficiency trade-off achieved by RL-Guided Sampling.

Method	AIME24			AIME25			HMMT25			Avg.		
	Acc. ↑	Seq. Tokens ↓	ToT. Tokens ↓	Acc. ↑	Seq. Tokens ↓	ToT. Tokens ↓	Acc. ↑	Seq. Tokens ↓	ToT. Tokens ↓	Acc. ↑	Seq. Tokens ↓	ToT. Tokens ↓
Model: Qwen3-0.6B-Thinking												
SC@32	22.6	24.7k	495.6k	30.1	23.5k	441.1k	16.4	23.8k	458.8k	23.0	24.0k	465.2k
ASC	22.6	437.1k	437.1k	30.1	359.5k	359.5k	16.4	349.4k	349.4k	23.0	382.0k (+1492.0%)	382.0k (-17.9%)
ESC	22.3	138.9k	480.3k	29.8	121.0k	415.2k	16.4	127.9k	436.9k	22.9	129.3k (+438.7%)	444.1k (-4.5%)
RL-guided	$22.6_{\pm 0.06}$	115.7k	356.0k	$30.0_{\pm 0.04}$	97.3k	298.7k	$16.4_{\pm 0.02}$	89.8k	272.7k	23.0	100.9k (+320.6%)	309.1k (-33.6%)
Model: Qwen3-1.7B-Thinking												
SC@32	68.3	26.2k	494.1k	44.1	25.8k	511.5k	25.9	27.7k	587.8k	46.1	26.6k	531.1k
ASC	68.2	317.7k	317.7k	44.2	358.8k	358.8k	26.0	420.6k	420.6k	46.1	365.7k (+1275.0%)	365.7k (-31.1%)
ESC	67.3	118.7k	412.8k	44.1	116.6k	421.7k	26.4	135.4k	505.8k	45.9	123.6k (+364.6%)	446.8k (-15.9%)
RL-guided	$67.6_{\pm 0.04}$	67.6k	210.2k	$44.6_{\pm 0.02}$	83.9k	271.6k	$26.7_{\pm 0.03}$	90.0k	296.0k	46.3	80.5k (+202.7%)	259.2k (-51.2%)
Model: Qwen3-4B-Instruct-Thinking												
SC@32	73.3	13.0k	221.3k	57.5	13.1k	215.3k	33.6	15.0k	244.1k	54.8	13.7k	226.9k
ASC	73.3	127.8k	127.8k	57.5	126.6k	126.6k	33.6	147.1k	147.1k	54.8	133.8k (+875.8%)	133.8k (-41.0%)
ESC	72.7	48.2k	161.5k	57.0	49.7k	167.3k	33.6	57.1k	189.7k	54.4	51.7k (+277.0%)	172.8k (-23.8%)
RL-guided	$73.0_{\pm 0.07}$	30.8k	90.1k	$57.1_{\pm 0.14}$	32.1k	96.2k	$33.6_{\pm 0.03}$	34.6k	101.2k	54.6	32.5k (+136.7%)	95.8k (-57.8%)
Model: GPT-4.1-nano												
SC@32	37.1	5.3k	92.7k	33.5	5.0k	86.5k	12.7	4.2k	80.7k	27.8	4.8k	86.6k
ASC	37.1	68.6k	68.6k	33.5	59.7k	59.7k	12.7	62.5k	62.5k	27.8	63.6k (+1219.0%)	63.6k (-26.6%)
ESC	36.9	24.1k	81.1k	33.2	22.4k	75.2k	12.3	20.9k	73.3k	27.5	22.5k (+365.7%)	76.5k (-11.7%)
RL-guided	$36.9_{\pm 0.04}$	26.0k	55.4k	$33.4_{\pm 0.05}$	23.0k	49.2k	$12.6_{\pm 0.02}$	22.8k	49.9k	27.7	24.0k (+396.7%)	51.5k (-40.6%)

Table 6: **Comparison of test-time scaling approaches across three benchmarks.** Acc. denotes accuracy, Seq. Tokens measures Sequential Tokens, and Tot. Tokens measures Total Tokens. We use the same parameter setup as in Table 1.

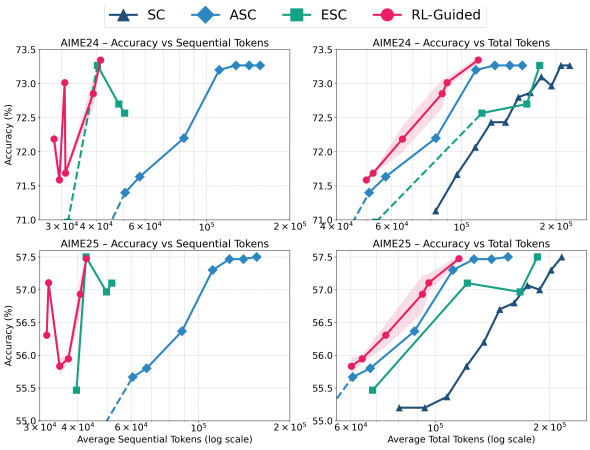

Figure 5: Accuracy–token scaling curves comparing the SC, ASC, ESC and RL-Guided Sampling. across different models and benchmarks. Results are generated with Qwen3-4B-Instruct on the AIME24 and AIME25 datasets.

B.2 Additional Details and Results for Scaling Analysis

For Figure 2, we conduct a parameter sweep for RL-Guided Sampling with $\lambda_{\text{lat}} = 0$ and $\lambda_{\text{comp}} \in \{0.001, 0.005, 0.0075, 0.015, 0.02, 0.03\}$. For ASC, we sweep the stopping threshold $C_{\text{threshold}} \in \{0.6, 0.65, 0.75, 0.8, 0.87, 0.92, 0.95, 0.98, 0.99\}$, and for ESC, we sweep the chunk size $K \in \{2, 3, 5, 7\}$. In addition to sample-level metrics, Figure 5 reports the corresponding token-level scaling curves, where Total Tokens measure computation cost and Sequential Tokens measure latency. We further provide scaling curves using Qwen3-0.6B as the LLM sampler in Figure 7.

B.3 Additional Results for Explanatory Analysis

In Section 4.3, we present the explanatory analysis on the DAPO subset, which contains more questions and therefore provides clearer visualization. For completeness, Figure 6 reports the corresponding analysis on the AIME24 dataset, where we observe a similar trend.

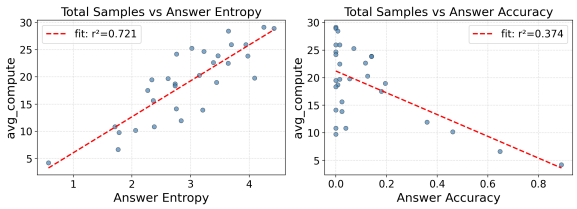

Figure 6: Correlation between total samples per query and Answer Entropy (left) alongside Answer Accuracy (right). Each point represents a distinct query from AIME24, with responses generated by Qwen3-0.6B.

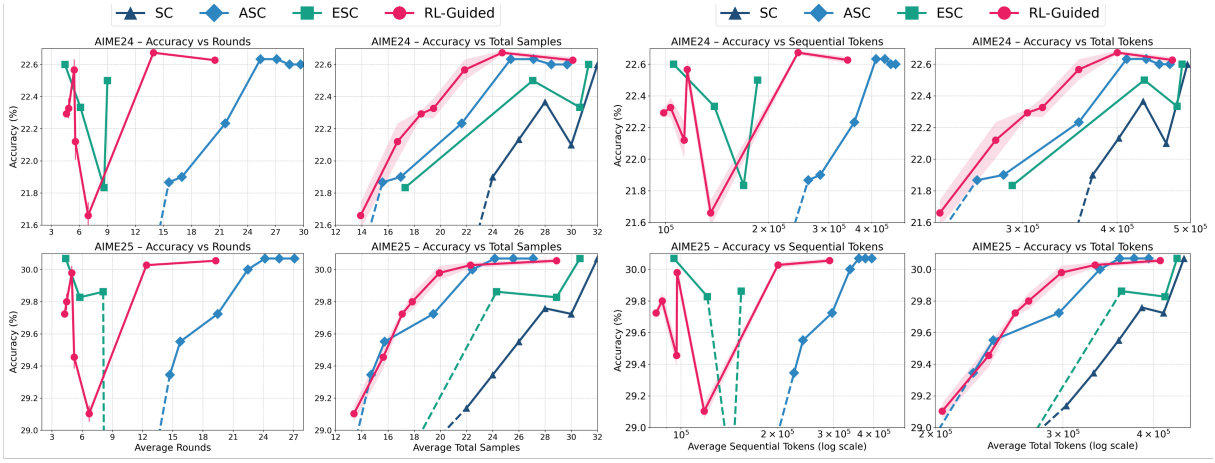

Figure 7: Accuracy–token scaling curves(right) and Accuracy–sampling scaling curves(left) comparing the SC, ASC, ESC and RL-Guided Sampling. across different models and benchmarks. Results are generated with Qwen3-0.6B on the AIME24 and AIME25 datasets.

C Proof of Proposition 1

This section provides the theoretical justification for Proposition 1, demonstrating the equivalence between our reinforcement learning objective and the Lagrangian relaxation of the budget-constrained adaptive sampling problem.

First, we clarify the definition of expected accuracy $J_{\text{ans}}(\pi_\theta)$ in the context of our MDP. Because our environment is intentionally decoupled from ground-truth labels, the terminal reward evaluates the final prediction against the maximum-budget consensus $y^\star$. Thus, $J_{\text{ans}}(\pi_\theta)$ mathematically represents the expected agreement with this asymptotic majority vote, serving as our proxy for accuracy. Furthermore, because $r^{\text{final}} \in \{1, -1\}$, its expectation is an affine transformation of the raw matching probability $P(\hat{y} = y^\star)$, specifically $2P - 1$. We define $J_{\text{ans}}(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[r^{\text{final}}_{t_{\text{stop}}}\right]$ to encapsulate this scaled objective.

In our proposed MDP, the controller's objective is to maximize the expected episodic return $J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$. Let t_{stop} denote the additional sampling steps taken before the episode terminates. The total return is the sum of rewards accumulated across all steps, $R(\tau) = \sum_{t=0}^{t_{\text{stop}}} r_t$.

By separating the terminal reward $r^{\text{final}}_{t_{\text{stop}}}$ from the intermediate step penalties incurred during generation, we can expand the expected return as follows:

$$\begin{aligned} J(\pi_\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[r^{\text{final}}_{t_{\text{stop}}} + \sum_{t=0}^{t_{\text{stop}}-1} (-\lambda_{\text{lat}} - \lambda_{\text{comp}} a_t)\right] \\ &= \mathbb{E}_{\tau\sim\pi_\theta}\left[r^{\text{final}}_{t_{\text{stop}}}\right] - \lambda_{\text{lat}}\mathbb{E}_{\tau\sim\pi_\theta}[t_{\text{stop}}] - \lambda_{\text{comp}}\mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{t_{\text{stop}}-1} a_t\right]. \end{aligned}$$

To strictly map these terms to the total metrics defined in Proposition 1, we must account for the initial observation round t_0 and the initial candidate pool size n_0 used to construct the initial state s_0. The expected total latency (sampling rounds) is $J_{\text{lat}}(\pi_\theta) = t_0 + \mathbb{E}_{\tau\sim\pi_\theta}[t_{\text{stop}}]$, and the expected total computation (generated samples) is $J_{\text{comp}}(\pi_\theta) = n_0 + \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{t_{\text{stop}}-1} a_t\right]$.

Substituting these relations back into the expanded return equation yields:

$$J(\pi_\theta) = J_{\text{ans}}(\pi_\theta) - \lambda_{\text{lat}}\big(J_{\text{lat}}(\pi_\theta) - t_0\big) - \lambda_{\text{comp}}\big(J_{\text{comp}}(\pi_\theta) - n_0\big).$$

Now, consider the budget-constrained adaptive sampling problem defined in Proposition 1, which seeks to maximize final expected accuracy proxy subject to predefined operational budgets:

$$\begin{aligned} \max_{\pi_\theta} \quad & J_{\text{ans}}(\pi_\theta) \\ \text{s.t.} \quad & J_{\text{lat}}(\pi_\theta) - C_{\text{lat}} \leqslant 0, \\ & J_{\text{comp}}(\pi_\theta) - C_{\text{comp}} \leqslant 0. \end{aligned}$$

By introducing non-negative dual variables $\lambda_{\text{lat}}, \lambda_{\text{comp}} \geqslant 0$ for the inequality constraints, the Lagrangian relaxation of this problem is:

$$\mathcal{L}(\pi_\theta, \lambda_{\text{lat}}, \lambda_{\text{comp}}) = J_{\text{ans}}(\pi_\theta) - \lambda_{\text{lat}}\big(J_{\text{lat}}(\pi_\theta) - C_{\text{lat}}\big) - \lambda_{\text{comp}}\big(J_{\text{comp}}(\pi_\theta) - C_{\text{comp}}\big).$$

Rearranging the Lagrangian to separate the policy-dependent components from the constant terms gives:

$$\mathcal{L}(\pi_\theta, \lambda_{\text{lat}}, \lambda_{\text{comp}}) = J_{\text{ans}}(\pi_\theta) - \lambda_{\text{lat}} J_{\text{lat}}(\pi_\theta) - \lambda_{\text{comp}} J_{\text{comp}}(\pi_\theta) + \lambda_{\text{lat}} C_{\text{lat}} + \lambda_{\text{comp}} C_{\text{comp}}.$$

Comparing this expression with our derived $J(\pi_\theta)$, we establish the exact algebraic relationship:

$$\mathcal{L}(\pi_\theta, \lambda_{\text{lat}}, \lambda_{\text{comp}}) = J(\pi_\theta) - \lambda_{\text{lat}} t_0 - \lambda_{\text{comp}} n_0 + \lambda_{\text{lat}} C_{\text{lat}} + \lambda_{\text{comp}} C_{\text{comp}}.$$

Because the scaling mappings and the offset terms (involving t_0, n_0, C_{lat}, and C_{comp}) are strictly affine transformations independent of the policy π_θ, maximizing the Lagrangian $\mathcal{L}$ with respect to π_θ is mathematically equivalent to maximizing our unconstrained RL objective $J(\pi_\theta)$. This explicitly confirms that optimizing the step-wise rewards in our defined MDP solves the Lagrangian relaxation of the constrained adaptive sampling problem. ■