RLVR from Within: Learning to Ask to Elicit Intrinsic Rewards for Test-Time Reasoning

Kuang-Da Wang[1,†] **Kai-Jie Lin**[1,†] **Guo-Xun Ko**[1] **Yu-Heng Lin**[1] **Yu Heng Hung**[1]
Travis M. Bartley[2] **Shuoyang Ding**[2] **Yu-Chiang Frank Wang**[23]
Chao-Han Huck Yang[2] **Guan-Ting Liu**[2] **Wen-Chih Peng**[1] **Ping-Chun Hsieh**[1]
[1]National Yang Ming Chiao Tung University, Hsinchu, Taiwan
[2]NVIDIA [3]National Taiwan University, Taipei, Taiwan

Abstract

Test-time reasoning for large language models (LLMs) often relies on external process reward models (PRMs) or value estimators to guide search over intermediate reasoning states. However, learning such rewards from final outcomes poses a challenging credit-assignment problem, and the reasoner's own self-verification may not reliably predict final-answer correctness. To address both challenges, we propose **AskR**, which learns *what to ask* to make the reasoner's self-verification more predictive of final correctness and thus more useful as an intrinsic reward. Rather than learning an external PRM, AskR learns a sub-question policy that structures reasoning into sequences of sub-questions and sub-answers (SubQAs), exposing dense self-verification signals at each SubQA. By shaping which intermediate reasoning states are exposed, AskR makes the induced intrinsic reward more predictive of final-answer correctness and therefore more useful for test-time optimization. Across our benchmarks, AskR consistently improves over PRM-guided reasoning under the same optimizer, by 7.12 points on average and up to 9.18 points depending on the optimizer. In its strongest setting, AskR further outperforms Reinforcement Learning with Verifiable Rewards (RLVR) on average, while using only 5% and 17% as much training data as RLVR on text and visual reasoning tasks, respectively. Our results suggest a new route to test-time reasoning: instead of learning how to evaluate arbitrary reasoning states, learn which states to expose so that the reasoner's existing verification signal becomes useful. Project page: https://rl-bandits-lab.github.io/AskR/.

1 Introduction

Recent advances (Lambert et al., 2024; Yue et al., 2025) in reinforcement learning with verifiable rewards (RLVR) have substantially improved the reasoning ability of large language models (LLMs). RLVR relies on externally verifiable final outcomes to reinforce successful reasoning trajectories, but these gains require expensive post-training of the backbone. This motivates a complementary line of work on *test-time reasoning*: improving solution quality by allocating additional computation at inference time rather than modifying the model parameters (Andrieu et al., 2003; Snell et al., 2025). Under this view, the central challenge is no longer how to train a stronger reasoner, but how to guide a fixed reasoner toward trajectories that are more likely to end in correct final answers.

A natural test-time analogue of RLVR is to extend its QA-level verifiability to intermediate reasoning: by structuring a reasoning trajectory as a sequence of sub-questions and sub-answers (SubQAs), each SubQA becomes an intermediate QA unit at which a verification signal can be obtained. Such

[†]Equal contribution. Correspondence to Kuang-Da Wang <gdwang.cs10@nycu.edu.tw>, Ping-Chun Hsieh <pinghsieh@nycu.edu.tw> and Guan-Ting Liu <dannliu@nvidia.com>.

Managing Agents that Manage Agents: Workshop on Responsible Use of Meta-Agents (NeurIPS 2026)

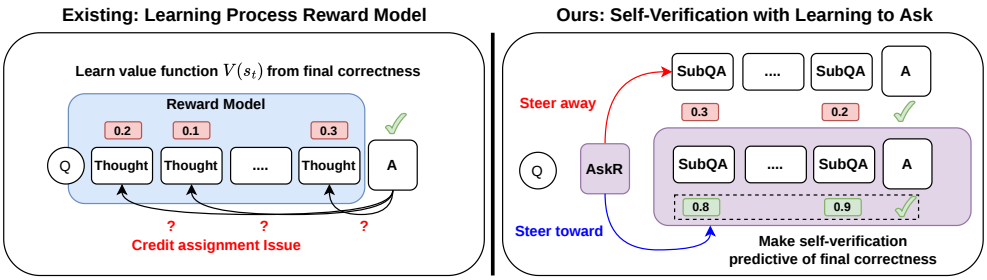


Figure 1: **From learning intermediate values to learning what to ask. Left:** Reward-guided test-time reasoning can score intermediate reasoning states, but learning such scores from final-answer correctness requires difficult credit assignment. **Right:** AskR instead structures reasoning into SubQAs so that the fixed reasoner can self-verify intermediate states, yielding dense intrinsic rewards that are more predictive of final correctness without training a separate PRM.

structured decomposition is also supported by prior work showing that sub-questioning can improve multi-step reasoning (Radhakrishnan et al., 2023; Shridhar et al., 2022). Following this view, we express a reasoning trajectory as a sequence of sub-questions and sub-answers (SubQAs), where each SubQA exposes an intermediate point at which the fixed reasoner can be evaluated. This converts the sparse outcome-reward view of RLVR into a sequence of dense intermediate signals that can potentially guide test-time optimization. The key challenge is not merely exposing intermediate states, but obtaining a reward over them that is predictive of final success.

Figure 1 contrasts two approaches to this problem. One possibility is to learn a process reward model (PRM) or value model, to estimate the value of intermediate reasoning states (Snell et al., 2025; Wang et al., 2025a; Puri et al., 2025; Wang et al., 2025c). Such dense supervision can guide test-time search, but learning reliable intermediate values from final outcomes is intrinsically difficult: final-answer correctness does not specify how credit should be assigned to individual states, and the same partial trajectory may lead to different outcomes depending on how reasoning continues (Cinquin et al., 2025; Pan et al., 2026; Wang et al., 2025a).

A second possibility is to use the reasoner's own self-verification signal. Prior work has shown that LLMs can critique or verify their own outputs (Madaan et al., 2023; Weng et al., 2023; Miao et al., 2024; Chen et al., 2025), making self-verification attractive as an intrinsic reward: it is available from the reasoner itself and requires no separately trained reward model. Yet self-verification alone is insufficient. A SubQA can be locally correct and confidently verified without making meaningful progress toward the correct final answer. Thus, the quantity needed for test-time optimization is not merely a signal of *local correctness*, but one that is *predictive of final-answer correctness*.

These two limitations suggest a different approach: rather than learning a better PRM, can we instead learn *which states to present to the self-verification procedure*? More specifically, can we control the sub-questions so that the reasoner's existing self-verification signal becomes informative about whether the overall trajectory will succeed? This leads to our central question:

> *Can we learn what to ask so that a reasoner's self-verification becomes a reliable intrinsic reward for test-time reasoning?*

To this end, we propose **AskR**, a learned *asker for reasoning* that generates sub-questions so that the reasoner's own self-verification signals become useful intrinsic rewards. We refer to the resulting reward-guided test-time reasoning pipeline as *AskR-guided reasoning*.

Given a problem, AskR structures reasoning into a sequence of SubQAs, exposing intermediate points at which the fixed reasoner can self-verify its reasoning. Rather than training a separate SubQA-based reward model, which would reintroduce the credit-assignment problem of assigning target values to intermediate states, AskR keeps the backbone reasoner and self-verification procedure fixed and learns only the SubQ generation policy using final-answer correctness. Importantly, AskR is not trained simply to make the original problem easier. Instead, learning what to ask steers reasoning toward SubQA trajectories where self-verification is more predictive of final correctness.

Table 1: **Positioning AskR among related paradigms.** AskR introduces learning to ask for structuring reasoning into SubQAs and eliciting intrinsic rewards from the reasoner, enabling test-time optimization without a separately trained reward model or updates to the reasoner backbone.

Paradigm	Structured Reasoning	No Separate Reward Model	Test-Time Optimization	No Reasoner Backbone Update
Model training or adaptation				
Self-rewarded post-training (Yuan et al., 2024; Zhang et al., 2025b)	–	✓	–	–
Joint reasoning–verification training (Zhang et al., 2025a)	–	–	✓	–
Test-time adaptation (Zuo et al., 2025; Wang et al., 2026c)	–	✓	✓	–
Frozen-backbone test-time reasoning				
Verifier guidance (Lightman et al., 2024; Wang et al., 2024a)	–	–	✓	✓
Prompted structured reasoning (Zhou et al., 2023; Wang et al., 2023a; Yao et al., 2023; Madaan et al., 2023)	✓	✓	–	✓
AskR (ours)	✓	✓	✓	✓

✓: characteristic of the paradigm; –: not a defining characteristic.

Our contributions are threefold:

- We formulate test-time reward elicitation as an internal analogue of RLVR: SubQA-structured reasoning exposes dense verification points, but the resulting reward is useful only when it is predictive of final correctness.
- We propose **AskR**, which avoids learning a general value model over arbitrary SubQA states and instead learns a SubQ policy that makes the reasoner's own self-verification a correctness-aligned intrinsic reward.
- We show the effectiveness of AskR. AskR consistently outperforms PRM-guided reasoning across different test-time optimizers, remains effective across model families, and transfers across reasoner scales with a fixed learned AskR. With TMPC, AskR further outperforms RLVR on average while requiring substantially less training data.

2 Related Work

Table 1 positions AskR among representative paradigms related to reasoning and test-time optimization. The key distinction is that learning to ask allows AskR to structure reasoning into SubQAs and elicit model-internal intrinsic rewards, enabling test-time optimization without training a separate reward model or updating the reasoner backbone.

2.1 Reasoning with Reinforcement Learning and Verifiable Rewards

Reinforcement learning with verifiable rewards (RLVR) has emerged as a central paradigm for improving LLM reasoning by optimizing task-level rewards derived from verifiable outcomes (Guo et al., 2025; Shao et al., 2024). Recent work has studied how to scale this paradigm through improved data construction and training strategies, including synthetic task generation (Lu et al., 2026; Liao et al., 2025), abstraction discovery (Qu et al., 2026), and reinforcement-style pretraining objectives (Hatamizadeh et al., 2026). Related approaches also let models generate their own training rewards (Yuan et al., 2024; Zhang et al., 2025b), jointly train reasoning and verification (Zhang et al., 2025a), or adapt model parameters directly on unlabeled test data (Zuo et al., 2025; Wang et al., 2026c). While these paradigms improve or adapt the reasoner itself, AskR keeps the reasoner frozen and instead learns how to elicit a useful intrinsic reward for test-time reasoning. Table 1 summarizes these distinctions and positions AskR relative to training-based and frozen-backbone reasoning paradigms.

2.2 Test-Time Reasoning and Verification

A broad line of work improves LLM reasoning at test time without updating model parameters. Prompting-based methods elicit intermediate reasoning through chain-of-thought (Wei et al., 2022), decomposition (Zhou et al., 2023; Wang et al., 2023a), or explicit search over candidate thoughts (Yao et al., 2023). Other approaches use self-refinement (Madaan et al., 2023; Yang et al., 2023), self-consistency (Wang et al., 2023b), or multi-agent deliberation (Du et al., 2024), while recent test-time optimization frameworks such as TPO and TMPC iteratively refine or search over candidate responses at inference time (Li et al., 2025; Wang et al., 2026a).

A closely related direction guides test-time search with externally learned process reward models, outcome reward models, or value models (Lightman et al., 2024; Yuan et al., 2025; Li and Li, 2025; Wang et al., 2025a; Puri et al., 2025; Snell et al., 2025; Wang et al., 2025c). These methods provide intermediate scores that can prioritize promising trajectories, but learning such scores requires assigning value to intermediate reasoning states from downstream outcomes. This credit-assignment problem becomes difficult over long horizons, and verifier errors can compound as search repeatedly optimizes imperfect intermediate estimates (Cinquin et al., 2025; Pan et al., 2026). LLMs can also critique, check, or verify their own outputs (Miao et al., 2024; Weng et al., 2023; Chen et al., 2025), providing a model-internal alternative that requires no separately trained verifier. However, self-verification alone need not indicate progress toward the correct final answer (Huang et al., 2024; Chen et al., 2024a). AskR connects these directions by learning *which intermediate states to elicit*, making the reasoner's existing self-verification more predictive of final-answer correctness.

2.3 Question Answering and Socratic Scaffolding

This work also connects to a long tradition in cognitive science and education showing that self-explanation and self-questioning can make intermediate reasoning explicit and support deeper comprehension (Chi et al., 1989, 1994; King, 1989; Vygotsky, 1978), with Socratic questioning framing reasoning as guided inquiry rather than direct answer production (Yang et al., 2005). Recent NLP work has operationalized related ideas through question decomposition (Radhakrishnan et al., 2023), Socratic sub-question generation (Shridhar et al., 2022), recursive self-questioning (Qi et al., 2023), learned reasoning behaviors such as verification and backtracking (Gandhi et al., 2025), and Socratic-style multimodal reasoning (Acuna et al., 2025).

AskR differs primarily in its objective.[1] Prior decomposition and Socratic methods use questions primarily to improve reasoning or final answers directly, whereas AskR learns questions to elicit a correctness-aligned intrinsic reward for test-time optimization.

3 Problem Formulation

We study *reward-guided test-time reasoning* for a fixed reasoner. Let $\mathcal{D}$ denote the underlying task distribution, with $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}_{i=1}^{N}$ as the training set and $\mathcal{D}_{\text{test}}$ as the held-out evaluation set. We assume access to an LLM reasoner π_θ, whose parameters remain fixed throughout both training and inference.

Given an input problem x, the reasoner produces a reasoning trajectory $\tau = (s_1, s_2, \ldots, s_T)$, where each s_t denotes an intermediate reasoning state. This notion is intentionally abstract: depending on the reasoning format, a state may correspond to a step, a sentence, a partial thought, or any other prefix-level unit over which reasoning operates. The trajectory τ ultimately yields a final answer $\hat{y}(\tau)$ through the reasoner π_θ, whose correctness is determined by a verifier $\mathcal{V}(\hat{y}, y) \in \{0, 1\}$.

In this study, the goal is to learn a reward function r_ϕ from $\mathcal{D}_{\text{train}}$ such that, when combined with a test-time optimizer $\mathcal{O}$, it guides the fixed reasoner π_θ toward reasoning trajectories that are more likely to yield correct final answers. Here, r_ϕ assigns rewards at the level of individual reasoning states or aggregated trajectories, depending on the optimizer and reasoning granularity. The optimizer $\mathcal{O}$ may take different forms, such as best-of-N (Lightman et al., 2024), tree search (Snell et al., 2025),

[1] Despite similar "learning to ask" terminology, interactive clarification methods (Wang et al., 2025b; Zhao et al., 2026; Wang et al., 2026b) address a different setting: the model asks an external user or environment to obtain missing information from an underspecified input. AskR instead asks internal sub-questions over a fixed problem and receives no additional task information through interaction.

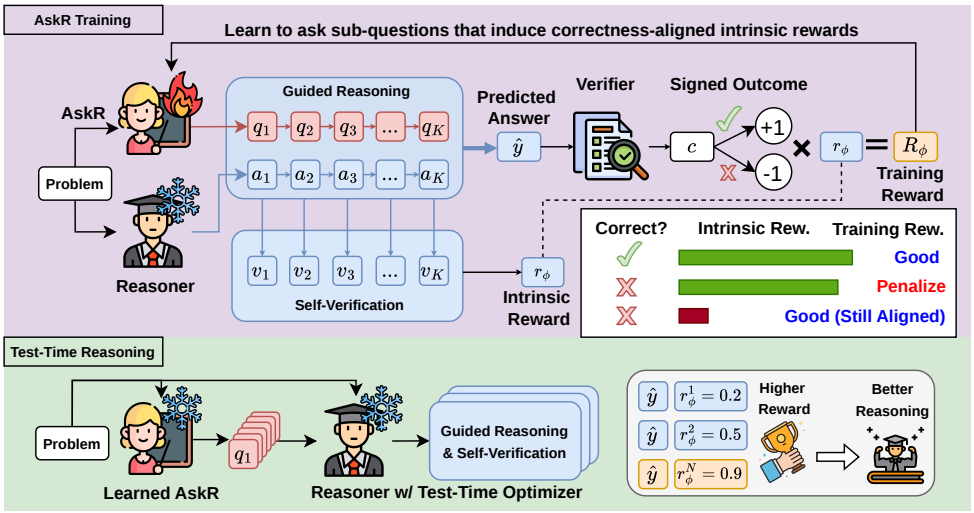


Figure 2: **Overview of AskR.** AskR (red) is responsible only for generating sub-questions, while the fixed reasoner (blue) produces sub-answers and self-verification scores. During training, the signed final outcome reinforces high intrinsic reward on correct trajectories and penalizes it on incorrect ones; at test time, this intrinsic reward guides test-time reasoning.

or trajectory optimization (Wang et al., 2026a; Li et al., 2025). It uses the learned reward to guide inference over candidate trajectories produced by π_θ. Formally, the quality of a learned reward is measured by the final-answer correctness it induces under test-time optimization:

$$r_\phi^\star = \arg\max_{r_\phi} \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{test}}} \left[\mathcal{V}\big(\hat{y}(\tau^\star), y\big)\right], \quad \text{where} \quad \tau^\star = \mathcal{O}(x, r_\phi; \pi_\theta). \tag{1}$$

This formulation separates three roles: the backbone reasoner π_θ generates candidate reasoning trajectories, the learned reward r_ϕ evaluates them, and the test-time optimizer $\mathcal{O}$ uses this reward to guide test-time reasoning.

4 Method

Figure 2 summarizes AskR. AskR is the learned asker g_ϕ, responsible only for generating sub-questions. The fixed reasoner π_θ produces the corresponding sub-answers, final prediction, and self-verification signals. During training, only g_ϕ is optimized; at test time, both g_ϕ and π_θ are frozen, and the elicited intrinsic reward guides an existing test-time optimizer.

4.1 Question Answering as Intrinsic Reward Elicitation

Given a problem x, AskR generates an ordered sequence of sub-questions

$$\mathcal{Q} = (q_1, \ldots, q_K), \qquad \mathcal{Q} \sim g_\phi(\cdot \mid x), \tag{2}$$

where q_t denotes the t-th generated sub-question, K is the number of sub-questions in the sampled sequence, and g_ϕ is the learned AskR policy. Conditioned on these questions, the fixed reasoner π_θ produces the corresponding sub-answers and final prediction in a structured response, which we write autoregressively as:

$$\mathcal{A} = (a_1, \ldots, a_K) \sim \pi_\theta(\cdot \mid x, \mathcal{Q}), \qquad \hat{y} \sim \pi_\theta(\cdot \mid x, \mathcal{Q}, \mathcal{A}), \tag{3}$$

where a_t is the reasoner's answer to q_t, and $\hat{y}$ is the final prediction synthesized from the resulting SubQA sequence.

This instantiates the abstract trajectory in Eq. 1 as $\tau = ((q_1, a_1), \ldots, (q_K, a_K))$, with state $s_t = (q_t, a_t)$. Given the complete trajectory τ, the same frozen reasoner π_θ is queried with a fixed self-verification prompt, denoted by f_{ver}, which jointly produces $(v_1, \ldots, v_K) \leftarrow f_{\text{ver}}(x, \tau)$, where $v_t \in \{0, 1\}$ indicates whether the corresponding SubQA is judged correct.

We aggregate these binary verification signals into the trajectory-level intrinsic reward

$$r_\phi(x, \tau) = \frac{1}{K} \sum_{t=1}^{K} v_t. \tag{4}$$

Although the self-verification procedure is fixed, we write r_ϕ to emphasize that the elicited reward depends on ϕ through the SubQA trajectory induced by g_ϕ; for a fixed trajectory τ, the self-verification procedure has no learnable parameters. The key challenge is that self-verification measures SubQA-level correctness: a high self-verification score need not imply progress toward the correct final answer. AskR therefore learns the SubQ generation policy so that the aggregate self-verification signal becomes more predictive of final-answer correctness.

4.2 Learning to Ask for Correctness-Aligned Intrinsic Rewards

AskR receives no ground-truth supervision on the sub-questions or individual SubQAs; its only ground-truth supervision is final-answer correctness.

Correctness-aligned training reward. For a training example (x, y), a sampled $\mathcal{Q}$ induces trajectory τ and final answer $\hat{y}$. Let $\mathcal{V}(\hat{y}, y) \in \{0, 1\}$ denote final-answer correctness, and define the signed outcome $c(\hat{y}, y) = 2\mathcal{V}(\hat{y}, y) - 1 \in \{-1, +1\}$. Our central training reward is

$$\boxed{R_\phi(x, y, \tau) = c(\hat{y}, y)\, r_\phi(x, \tau)} \tag{5}$$

For a trajectory yielding a correct final answer, high intrinsic reward is reinforced; for one yielding an incorrect final answer, the same high intrinsic reward is explicitly penalized.

AskR is therefore *not* trained merely to maximize self-verification. Instead, maximizing R_ϕ learns which SubQA trajectories to elicit so that high intrinsic reward is associated with correct final answers and suppressed on incorrect ones, making the fixed self-verification signal predictive of final correctness without SubQA-level target values.

Optimizing AskR. We optimize the AskR policy g_ϕ with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For each training example (x, y), the rollout policy $g_{\phi_{\text{old}}}$ samples a group of G questioning sequences $\{\mathcal{Q}_i\}_{i=1}^{G}$, where $\mathcal{Q}_i = (q_{i,1}, \ldots, q_{i,K_i})$ contains K_i sub-questions. Each $\mathcal{Q}_i$ induces a SubQA trajectory τ_i, final prediction $\hat{y}_i$, and sequence-level reward

$$R_i = R_\phi(x, y, \tau_i) = c(\hat{y}_i, y)\, r_\phi(x, \tau_i).$$

We normalize rewards within the group to obtain

$$A_i = \frac{R_i - \mu_R}{\sigma_R + \delta}, \qquad \mu_R = \frac{1}{G} \sum_{j=1}^{G} R_j, \qquad \sigma_R^2 = \frac{1}{G} \sum_{j=1}^{G} (R_j - \mu_R)^2, \tag{6}$$

where $\delta > 0$ is a small constant for numerical stability. Let $\mathbf{o}_i = (o_{i,1}, \ldots, o_{i,L_i})$ denote the token sequence realizing $\mathcal{Q}_i$, with length L_i. AskR is updated at the token level using the GRPO clipped objective

$$\mathcal{L}_{\text{AskR}}(\phi) = -\mathbb{E}\left[\frac{1}{\sum_{i=1}^{G} L_i} \sum_{i=1}^{G} \sum_{\ell=1}^{L_i} \min(\rho_{i,\ell} A_i,\ \text{clip}(\rho_{i,\ell})\, A_i) \right], \tag{7}$$

where

$$\rho_{i,\ell} = \frac{g_\phi(o_{i,\ell} \mid x, o_{i,<\ell})}{g_{\phi_{\text{old}}}(o_{i,\ell} \mid x, o_{i,<\ell})} \tag{8}$$

is the token-level importance ratio. Here, ℓ indexes tokens in the AskR output, whereas K_i indexes SubQs in the structured questioning sequence. Thus, SubQA-level self-verification determines the sequence-level reward and advantage, while GRPO updates AskR through token-level policy likelihoods. Exact optimization hyperparameters and clipping details are provided in Appendix E.3.4.

4.3 Using Correctness-Aligned Intrinsic Rewards at Test Time

At test time, both g_ϕ and π_θ are frozen. For a problem x, AskR generates the SubQ sequence $\mathcal{Q}$, and the fixed reasoner produces the corresponding SubQA trajectory τ. The trajectory is scored by $r_\phi(x, \tau)$ using Eq. 4. The task verifier $\mathcal{V}$ used during training is no longer required; the test-time optimizer directly uses the elicited reward: $\tau^\star = \mathcal{O}(x, r_\phi; \pi_\theta)$ which instantiates Eq. 1.

5 Experiments

5.1 Experimental Setup

Benchmarks. We report main results on three language-only reasoning benchmarks, MMLU-Pro (Wang et al., 2024b), LAB-Bench (Laurent et al., 2024), and SciEval (Sun et al., 2024), and three vision-language reasoning benchmarks, V* Bench (Wu and Xie, 2024), MMStar (Chen et al., 2024b), and CV-Bench (Tong et al., 2024). Details of the benchmark composition are in Appendix D.

Model Backbones. We use `Qwen2.5-7B-Instruct` (Qwen et al., 2025) as the backbone reasoner for language-only experiments and `Qwen2.5-VL-7B-Instruct` (Bai et al., 2025) for vision-language experiments. The backbone reasoner is frozen for all test-time methods.

Baselines and Test-Time Optimization Methods. We compare four categories of methods: prompting-based methods (CoT (Wei et al., 2022), ToT (Yao et al., 2023), Least-to-Most (Zhou et al., 2023), Self-Refine (Madaan et al., 2023)), AskR-guided test-time reasoning, PRM-guided test-time reasoning, and value-model-guided test-time reasoning. For AskR and PRM rewards, we instantiate three optimizers: Best-of-N (BoN), TPO (Li et al., 2025), and TMPC (Wang et al., 2026a). We report value-model-guided search using VGS (Wang et al., 2025a) and PF (Puri et al., 2025), since value models represent a different form of external guidance from process reward models. We further compare against RLVR as a training-based baseline. For a fair comparison, we use the same candidate-generation budget (i.e., search budget) across methods. Implementation details and token-accuracy analysis are provided in Appendix E and Appendix C, respectively.

Training Setup and Evaluation. For language-only experiments, AskR is trained with GRPO on 5K STEM and verifiable mathematics examples from SYNTHETIC-1 (Mattern et al., 2025). For vision-language experiments, AskR is trained with GRPO on 5K verifiable examples from LONGPERCEPTUALTHOUGHTS (Liao et al., 2025). In both settings, AskR is trained without step-wise question labels or gold sub-question decompositions; it only observes final-answer correctness. For RLVR comparison, the language-only baseline is first supervised fine-tuned on a 100K DeepSeek reasoning-trace subset from SYNTHETIC-1, and is then further optimized with GRPO on 5K examples from the same training set used for AskR. The vision-language baseline is trained similarly: it is first supervised fine-tuned on 25K reasoning traces from LONGPERCEPTUALTHOUGHTS, followed by GRPO on 5K examples, again using the same training set as AskR. All results are averaged over 8 independent sampling trials per problem.

5.2 Does Learning to Ask Improve Test-Time Reasoning?

Table 2 reports results across six language-only and vision-language benchmarks, highlighting three main observations:

(1) Prompting alone provides limited improvements. Prompting-based approaches yield modest and sometimes inconsistent gains, especially on vision-language benchmarks. This suggests that additional test-time computation alone is insufficient; effective reasoning also requires a reliable signal to guide that computation.

(2) AskR-guided reasoning outperforms PRM guidance under matched optimizers. Under matched optimizers, replacing PRM guidance with AskR consistently improves both language-only and vision-language performance, by **+5.63**, **+6.55**, and **+9.18** points under BoN, TPO, and TMPC, respectively. Even with simple BoN, AskR surpasses the strongest value-model-guided baseline, VM + PF, by **+2.94** points on average. The gap also persists with a larger PRM training budget: Appendix F.3 shows that PRM remains behind AskR when trained on 30K examples, six times the AskR training budget.

Table 2: Main results on the LLM/VLM benchmarks (mean accuracy over 8 independent trials; an expanded table with standard deviations is provided in Appendix F). AskR rows are highlighted in blue. Bold denotes the best in each column, and underlining denotes the second best.

	Language-only				Vision-language			
Method	MMLU-Pro	LAB-Bench	SciEval	**Avg.**	V* Bench	MMStar	CV-Bench	**Avg.**
Training-based reference								
RLVR	**64.19**	33.75	<u>68.78</u>	<u>55.57</u>	<u>79.50</u>	60.93	72.91	<u>71.11</u>
Prompting-based								
CoT	55.75	32.63	48.75	45.71	72.50	59.13	68.47	66.70
L2M	56.70	29.80	43.23	43.24	72.94	55.21	68.96	65.70
ToT	54.00	33.72	43.27	43.66	75.06	60.53	66.25	67.28
Self-Refine	57.21	33.53	59.53	50.09	73.53	57.00	66.34	65.62
Value-model (VM)-guided methods								
VM + VGS	55.36	32.20	48.70	45.42	71.24	61.83	<u>74.60</u>	69.22
VM + PF	61.03	31.89	51.99	48.30	78.14	61.05	71.74	70.31
BoN (matched optimizer)								
PRM + BoN	58.40	30.49	49.67	46.19	72.75	58.80	69.54	67.03
AskR + BoN	61.23	<u>35.05</u>	65.43	53.90	77.48	61.73	72.53	70.58
TPO (matched optimizer)								
PRM + TPO	56.18	29.70	48.50	44.79	78.01	42.66	72.46	64.38
AskR + TPO	60.61	33.19	63.10	52.30	76.43	<u>62.40</u>	71.07	69.97
TMPC (matched optimizer)								
PRM + TMPC	60.30	31.17	47.86	46.44	75.78	58.40	66.98	67.05
AskR + TMPC	<u>63.76</u>	**37.29**	**70.45**	**57.17**	**84.37**	**63.23**	**76.43**	**74.68**

Table 3: Performance of AskR before and after training across BoN, TPO, and TMPC (section 4 training). Small numbers in brackets denote the change relative to trained AskR.

		Language-only				Vision-language			
Optimizer	**AskR**	**MMLU-Pro**	**LAB-Bench**	**SciEval**	**Avg.**	**V* Bench**	**MMStar**	**CV-Bench**	**Avg.**
BoN	After Training	**61.23**	**35.05**	**65.43**	**53.90**	**77.48**	**61.73**	**72.53**	**70.58**
	Before Training	60.06 (-1.17)	34.52 (-0.53)	64.15 (-1.28)	52.91 (-0.99)	75.82 (-1.66)	58.83 (-2.90)	71.57 (-0.96)	68.74 (-1.84)
TPO	After Training	60.61	**33.19**	63.10	**52.30**	**76.43**	**62.40**	**71.07**	**69.97**
	Before Training	**60.64** (+0.03)	32.06 (-1.13)	**63.86** (+0.76)	52.19 (-0.11)	74.94 (-1.49)	59.94 (-2.46)	69.57 (-1.50)	68.15 (-1.82)
TMPC	After Training	**63.76**	**37.29**	**70.45**	**57.17**	**84.37**	**63.23**	**76.43**	**74.68**
	Before Training	59.23 (-4.53)	33.43 (-3.86)	66.53 (-3.92)	53.06 (-4.10)	80.65 (-3.72)	60.96 (-2.27)	72.53 (-3.90)	71.38 (-3.30)

(3) Stronger test-time optimization allows AskR-guided reasoning to surpass RLVR. When paired with TMPC, AskR outperforms RLVR by **+2.58** points on average across the six benchmarks, despite leaving the backbone reasoner frozen and requiring substantially less training data. This shows that AskR's coupled trajectory structuring and intrinsic-reward guidance can redirect test-time computation effectively enough to compete with, and even exceed, training-based improvements to the reasoner itself.

5.3 Does Learning to Ask Matter Across Test-Time Optimizers?

We next assess the contribution of learning the questioning policy by comparing AskR after training against the same question generator before AskR-specific training[2] while keeping the rest of the test-time pipeline unchanged. Table 3 shows that training improves average performance under all three optimizers, with consistent gains under BoN and the largest benefit under TMPC; gains under TPO are smaller and more variable. These results indicate that AskR's gains do not arise merely from introducing a SubQA interface or additional self-verification at inference time. Rather, learning what to ask jointly shapes the SubQA trajectories and the induced intrinsic reward used for test-time optimization, with the benefit becoming especially pronounced for stronger optimizers. Illustrative examples of AskR on text and vision reasoning are provided in Appendix G.

[2] *AskR (Before Training)* denotes the pretrained question-generator initialization before AskR-specific GRPO training, with the same SubQA interface and test-time pipeline.

Table 4: Generalization across model families. AskR consistently outperforms CoT and RLVR on both Qwen3-4B and Gemma3-4B, demonstrating generalization across model families.

		Language-only				Vision-language			
Backbone	**Method**	**MMLU-Pro**	**LAB-Bench**	**SciEval**	**Avg.**	**V* Bench**	**MMStar**	**CV-Bench**	**Avg.**
Qwen3-4B	CoT	69.60	34.98	71.60	58.73	82.72	66.00	58.50	69.07
	AskR + TMPC	**78.42**	**44.83**	**76.46**	**66.57**	**84.50**	**70.33**	**66.00**	**73.61**
	RLVR	73.98	36.50	76.15	62.21	80.10	68.50	59.85	69.48
Gemma3-4B	CoT	41.24	26.50	58.50	42.08	36.65	52.50	27.50	38.88
	AskR + TMPC	**47.45**	**32.73**	**64.62**	**48.27**	**46.21**	**58.31**	**34.50**	**46.34**
	RLVR	45.92	26.00	63.50	45.14	37.17	54.50	26.00	39.22

5.4 Does AskR Generalize Across Model Families and Scales?

Table 2 uses Qwen2.5-7B as the backbone, raising the question of whether the effectiveness of AskR depends on a particular model family or model scale. We examine these two dimensions separately:

Generalization across model families. Table 4 evaluates AskR on Qwen3-4B and Gemma3-4B, testing whether the effectiveness of our approach extends beyond the Qwen2.5 backbone used in the main experiments. AskR + TMPC consistently outperforms both CoT and RLVR across all six benchmarks for both backbones. Relative to CoT, AskR improves the average performance from 63.90 to 70.09 on Qwen3-4B (**+6.19** points) and from 40.48 to 47.30 on Gemma3-4B (**+6.82** points). It also exceeds RLVR by **+4.24** and **+5.12** points on Qwen3-4B and Gemma3-4B, respectively. These results indicate that the benefits of learning to elicit intrinsic rewards are not specific to the Qwen2.5 backbone and extend to both a newer Qwen generation and a different model family.

Generalization across model scales. We next examine whether a single learned AskR remains effective as the backbone reasoner scales. Keeping AskR fixed to Qwen3-4B, we vary the Qwen3 reasoner from 1.7B to 32B, and also report AskR before training to isolate the effect of training. Figure 3 reports the average accuracy across MMLU-Pro, LAB-Bench, and SciEval; per-benchmark results are in Appendix F.1 (Table 6). AskR + TMPC improves over CoT at every scale, from **+9.96** points at 1.7B to **+4.70** at 32B, whereas AskR before training gains at most +1.69, showing that the improvement comes from the learned questioning policy rather than the SubQA interface alone. An AskR trained at small scale thus transfers to substantially larger reasoners, with diminishing but consistently positive benefit.

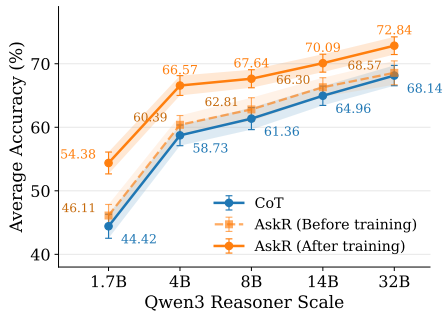


Figure 3: Performance of CoT and AskR before and after training under TMPC when keeping the AskR fixed to Qwen3-4B and varying only the reasoner scale.

5.5 Does AskR Training Align Intrinsic Reward with Final Correctness?

A useful intrinsic reward should be predictive of final-answer correctness. On all evaluation examples used in our experiments for language-only reasoning, we measure Pearson correlation, Spearman correlation, and AUROC between the intrinsic reward $r_\phi(x, \tau)$ and final-answer correctness $\mathcal{V}(\hat{y}, y)$ before and after AskR training. Figure 4 reports the macro-average across the three benchmarks, showing consistent improvements. Appendix F.2 provides the corresponding per-benchmark before-and-after results and further shows that the improvement arises from the correctness-aligned objective, rather than simply training AskR to ask questions that make the original task easier.

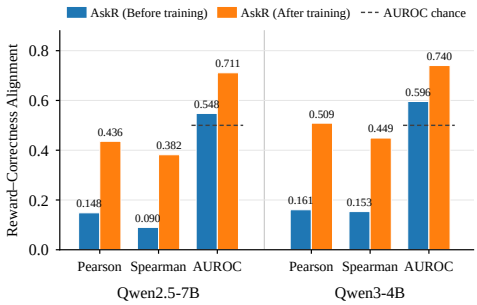


Figure 4: Reward–correctness alignment before and after AskR training.

6 Conclusion

We introduced **AskR**, a learning-to-ask framework for eliciting useful intrinsic rewards from a fixed LLM reasoner. Rather than learning an external reward model over intermediate states, AskR structures reasoning into SubQAs and learns which questions to ask so that the reasoner's own self-verification becomes more predictive of final-answer correctness. Our results show that this correctness-aligned intrinsic reward consistently improves test-time reasoning across multiple optimizers, remains effective across model families, transfers across reasoner scales with a fixed learned AskR. More broadly, AskR suggests a complementary route to stronger test-time reasoning: instead of learning a better reasoner or reward model, learn how to expose reasoning so that the model's existing verification capability becomes useful for guiding computation.

Reproducibility statement

The methodology is described in Section 4. All training datasets and evaluation benchmarks used in our experiments are publicly available. The experimental setup is summarized in Section 5.1, with detailed descriptions of the training datasets, benchmark composition, and evaluation settings provided in Appendix D. We provide comprehensive implementation details, including prompt templates, training hyperparameters, baseline configurations, and inference procedures for the test-time optimizers, in Appendix E.

References

Acuna, D., Lu, X., Jung, J., Kim, H., Kar, A., Fidler, S., Choi, Y., 2025. Socratic-mcts: Test-time visual reasoning by asking the right questions, in: EMNLP, pp. 24147–24160. URL: https://doi.org/10.18653/v1/2025.emnlp-main.1230.

Andrieu, C., De Freitas, N., Doucet, A., Jordan, M.I., 2003. An introduction to mcmc for machine learning. Machine learning 50, 5–43.

Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., Lin, J., 2025. Qwen2.5-vl technical report. URL: https://arxiv.org/abs/2502.13923, arXiv:2502.13923.

Chen, A., Phang, J., Parrish, A., Padmakumar, V., Zhao, C., Bowman, S.R., Cho, K., 2024a. Two failures of self-consistency in the multi-step reasoning of LLMs. Transactions on Machine Learning Research URL: https://openreview.net/forum?id=5nBqY1y96B.

Chen, J., Ren, J., Chen, X., Yang, C., Sun, R., Yoon, J., Arik, S.O., 2025. SETS: Leveraging self-verification and self-correction for improved test-time scaling. Transactions on Machine Learning Research URL: https://openreview.net/forum?id=Wv9NMJoKww.

Chen, L., Li, J., Dong, X., Zhang, P., Zang, Y., Chen, Z., Duan, H., Wang, J., Qiao, Y., Lin, D., Zhao, F., 2024b. Are we on the right way for evaluating large vision-language models?, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=evP9mxNNxJ.

Chi, M.T., Bassok, M., Lewis, M.W., Reimann, P., Glaser, R., 1989. Self-explanations: How students study and use examples in learning to solve problems. Cognitive science 13, 145–182.

Chi, M.T., De Leeuw, N., Chiu, M.H., LaVancher, C., 1994. Eliciting self-explanations improves understanding. Cognitive science 18, 439–477.

Cinquin, T., Pleiss, G., Kristiadi, A., 2025. Limits of PRM-guided tree search for mathematical reasoning with LLMs, in: The 5th Workshop on Mathematical Reasoning and AI at NeurIPS 2025. URL: https://openreview.net/forum?id=0wHkPbS2T8.

Du, Y., Li, S., Torralba, A., Tenenbaum, J.B., Mordatch, I., 2024. Improving factuality and reasoning in language models through multiagent debate, in: Forty-first International Conference on Machine Learning. URL: https://openreview.net/forum?id=zj7YuTE4t8.

Gandhi, K., Chakravarthy, A.K., Singh, A., Lile, N., Goodman, N., 2025. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective STars, in: Second Conference on Language Modeling. URL: https://openreview.net/forum?id=QGJ9ttXLTy.

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z.F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Ding, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Chen, J., Yuan, J., Tu, J., Qiu, J., Li, J., Cai, J.L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., You, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Zhou, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R.J., Jin, R.L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S.S., Zhou, S., Wu, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W.L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X.Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y.K., Wang, Y.Q., Wei, Y.X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y.X., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z.Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., Zhang, Z., 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. Nature 645, 633–638. URL: http://dx.doi.org/10.1038/s41586-025-09422-z, doi:10.1038/s41586-025-09422-z.

Han, D., Han, M., 2023. Unsloth: Open-source framework for running and training models. URL: https://github.com/unslothai/unsloth.

Hatamizadeh, A., Akter, S.N., Prabhumoye, S., Kautz, J., Patwary, M., Shoeybi, M., Catanzaro, B., Choi, Y., 2026. RLP: Reinforcement as a pretraining objective, in: The Fourteenth International Conference on Learning Representations. URL: https://openreview.net/forum?id=9Gp45bnDrJ.

Huang, J., Chen, X., Mishra, S., Zheng, H.S., Yu, A.W., Song, X., Zhou, D., 2024. Large language models cannot self-correct reasoning yet, in: The Twelfth International Conference on Learning Representations. URL: https://openreview.net/forum?id=IkmD3fKBPQ.

King, A., 1989. Effects of self-questioning training on college students' comprehension of lectures. Contemporary Educational Psychology 14, 366–381.

Lambert, N., Morrison, J., Pyatkin, V., Huang, S., Ivison, H., Brahman, F., Miranda, L.J.V., Liu, A., Dziri, N., Lyu, S., et al., 2024. Tulu 3: Pushing frontiers in open language model post-training. arXiv preprint arXiv:2411.15124 .

Laurent, J.M., Janizek, J.D., Ruzo, M., Hinks, M.M., Hammerling, M.J., Narayanan, S., Ponnapati, M., White, A.D., Rodriques, S.G., 2024. Lab-bench: Measuring capabilities of language models for biology research. URL: https://arxiv.org/abs/2407.10362, arXiv:2407.10362.

Li, W., Li, Y., 2025. Process reward model with q-value rankings, in: The Thirteenth International Conference on Learning Representations. URL: https://openreview.net/forum?id=wQEdh2cgEk.

Li, Y., Hu, X., Qu, X., Li, L., Cheng, Y., 2025. Test-time preference optimization: On-the-fly alignment via iterative textual feedback, in: Forty-second International Conference on Machine Learning. URL: https://openreview.net/forum?id=ArifAHrEVD.

Liao, Y.H., Elflein, S., He, L., Leal-Taixé, L., Choi, Y., Fidler, S., Acuna, D., 2025. Longperceptualthoughts: Distilling system-2 reasoning for system-1 perception, in: Second Conference on Language Modeling. URL: https://openreview.net/forum?id=SrKdi4MsUW.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., Cobbe, K., 2024. Let's verify step by step, in: The Twelfth International Conference on Learning Representations. URL: https://openreview.net/forum?id=v8L0pN6EOi.

Lu, X., Acuna, D., Jung, J., Hu, J., Zhang, D., Diao, S., Zou, Y., Zhang, S., Cui, B., Liu, M., Kim, H., Ammanabrolu, P., Kautz, J., Dong, Y., Choi, Y., 2026. Golden goose: A simple trick to synthesize unlimited rlvr tasks from unverifiable internet text. URL: https://arxiv.org/abs/2601.22975, arXiv:2601.22975.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B.P., Hermann, K., Welleck, S., Yazdanbakhsh, A., Clark, P., 2023. Self-refine: Iterative refinement with self-feedback, in: Thirty-seventh Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=S37hOerQLB.

Mattern, J., Jaghouar, S., Basra, M., Straube, J., Ferrante, M.D., Gabriel, F., Ong, J.M., Weisser, V., Hagemann, J., 2025. Synthetic-1: Two million collaboratively generated reasoning traces from deepseek-r1. URL: https://www.primeintellect.ai/blog/synthetic-1-release.

Miao, N., Teh, Y.W., Rainforth, T., 2024. Selfcheck: Using LLMs to zero-shot check their own step-by-step reasoning, in: The Twelfth International Conference on Learning Representations. URL: https://openreview.net/forum?id=pTHfApDakA.

Pan, P.C., Liang, Y., Lin, S., 2026. Reward modeling for reinforcement learning-based llm reasoning: Design, challenges, and evaluation. URL: https://arxiv.org/abs/2602.09305, arXiv:2602.09305.

Puri, I., Sudalairaj, S., Xu, G., Bhandwaldar, A., Xu, K., Srivastava, A., 2025. Rollout roulette: A probabilistic inference approach to inference-time scaling of LLMs using particle-based monte carlo methods, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=qPQUrjiA0q.

Qi, J., Xu, Z., Shen, Y., Liu, M., Jin, D., Wang, Q., Huang, L., 2023. The art of SOCRATIC QUESTIONING: Recursive thinking with large language models, in: Bouamor, H., Pino, J., Bali, K. (Eds.), Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Singapore. pp. 4177–4199. URL: https://aclanthology.org/2023.emnlp-main.255/, doi:10.18653/v1/2023.emnlp-main.255.

Qu, Y., Singh, A., Lee, Y., Setlur, A., Salakhutdinov, R., Finn, C., Kumar, A., 2026. RLAD: Training LLMs to discover abstractions for solving reasoning problems, in: The Fourteenth International Conference on Learning Representations. URL: https://openreview.net/forum?id=fvJPjCioeR.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., Qiu, Z., 2025. Qwen2.5 technical report. URL: https://arxiv.org/abs/2412.15115, arXiv:2412.15115.

Radhakrishnan, A., Nguyen, K., Chen, A., Chen, C., Denison, C., Hernandez, D., Durmus, E., Hubinger, E., Kernion, J., Lukošiūtė, K., Cheng, N., Joseph, N., Schiefer, N., Rausch, O., McCandlish, S., Showk, S.E., Lanham, T., Maxwell, T., Chandrasekaran, V., Hatfield-Dodds, Z., Kaplan, J., Brauner, J., Bowman, S.R., Perez, E., 2023. Question decomposition improves the faithfulness of model-generated reasoning. URL: https://arxiv.org/abs/2307.11768, arXiv:2307.11768.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y.K., Wu, Y., Guo, D., 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. URL: https://arxiv.org/abs/2402.03300, arXiv:2402.03300.

Shridhar, K., Macina, J., El-Assady, M., Sinha, T., Kapur, M., Sachan, M., 2022. Automatic generation of socratic subquestions for teaching math word problems, in: Goldberg, Y., Kozareva, Z., Zhang, Y. (Eds.), Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates. pp. 4136–4149. URL: https://aclanthology.org/2022.emnlp-main.277/, doi:10.18653/v1/2022.emnlp-main.277.

Snell, C.V., Lee, J., Xu, K., Kumar, A., 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning, in: The Thirteenth International Conference on Learning Representations. URL: https://openreview.net/forum?id=4FWAwZtd2n.

Sun, L., Han, Y., Zhao, Z., Ma, D., Shen, Z., Chen, B., Chen, L., Yu, K., 2024. Scieval: a multi-level large language model evaluation benchmark for scientific research, in: Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence, AAAI Press. URL: https://doi.org/10.1609/aaai.v38i17.29872, doi:10.1609/aaai.v38i17.29872.

Tong, S., II, E.L.B., Wu, P., Woo, S., IYER, A.J., Akula, S.C., Yang, S., Yang, J., Middepogu, M., Wang, Z., Pan, X., Fergus, R., LeCun, Y., Xie, S., 2024. Cambrian-1: A fully open, vision-centric exploration of multimodal LLMs, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=Vi8AepAXGy.

Vygotsky, L.S., 1978. Mind in society: The development of higher psychological processes. volume 86. Harvard university press.

Wang, K., Zhou, J.P., Chang, J.D., Gao, Z., Kallus, N., Brantley, K., Sun, W., 2025a. Value-guided search for efficient chain-of-thought reasoning, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=jOsuKwiCL0.

Wang, K.D., Chen, T.R., Hung, Y.H., Ko, G.X., Ding, S., Wu, Y.H., Wang, Y.C.F., Yang, C.H.H., Peng, W.C., Hsieh, P.C., 2026a. Test-time alignment for large language models via textual model predictive control, in: The Fourteenth International Conference on Learning Representations. URL: https://openreview.net/forum?id=DsS3xRPSs5.

Wang, K.D., Wang, Z., Wang, W.Y., Shimose, Y., Kim, J., Takamatsu, S., 2026b. Agentic model predictive questioning control in visual design, in: Forty-third International Conference on Machine Learning. URL: https://openreview.net/forum?id=hvrnTzLyM7.

Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R.K.W., Lim, E.P., 2023a. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, in: Rogers, A., Boyd-Graber, J., Okazaki, N. (Eds.), Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Toronto, Canada. pp. 2609–2634. URL: https://aclanthology.org/2023.acl-long.147/, doi:10.18653/v1/2023.acl-long.147.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., Sui, Z., 2024a. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations, in: Ku, L.W., Martins, A., Srikumar, V. (Eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Bangkok, Thailand. pp. 9426–9439. URL: https://aclanthology.org/2024.acl-long.510/, doi:10.18653/v1/2024.acl-long.510.

Wang, R., Huang, W., Cao, Q., Iwasawa, Y., Matsuo, Y., Guo, J., 2026c. SELF-HARMONY: LEARNING TO HARMONIZE SELF-SUPERVISION AND SELF-PLAY IN TEST-TIME REINFORCEMENT LEARNING, in: The Fourteenth International Conference on Learning Representations. URL: https://openreview.net/forum?id=ZzG6oJ5ehI.

Wang, W., Juluan, S., Ling, Z., Chan, Y.K., Wang, C., Lee, C., Yuan, Y., Huang, J.t., Jiao, W., Lyu, M.R., 2025b. Learning to ask: When LLM agents meet unclear instruction, in: Christodoulopoulos, C., Chakraborty, T., Rose, C., Peng, V. (Eds.), Proceedings of the 2025

Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Suzhou, China. pp. 21773–21784. URL: https://aclanthology.org/2025.emnlp-main.1104/, doi:10.18653/v1/2025.emnlp-main.1104.

Wang, X., Li, Y., Feng, S., Yuan, P., Zhang, Y., Shi, J., Tan, C., Pan, B., Hu, Y., Li, K., 2025c. Every rollout counts: Optimal resource allocation for efficient test-time scaling, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=xSHqNf5Pdc.

Wang, X., Wei, J., Schuurmans, D., Le, Q.V., Chi, E.H., Narang, S., Chowdhery, A., Zhou, D., 2023b. Self-consistency improves chain of thought reasoning in language models, in: The Eleventh International Conference on Learning Representations. URL: https://openreview.net/forum?id=1PL1NIMMrw.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., Li, T., Ku, M., Wang, K., Zhuang, A., Fan, R., Yue, X., Chen, W., 2024b. MMLU-pro: A more robust and challenging multi-task language understanding benchmark, in: The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track. URL: https://openreview.net/forum?id=y10DM6R2r3.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., brian ichter, Xia, F., Chi, E.H., Le, Q.V., Zhou, D., 2022. Chain of thought prompting elicits reasoning in large language models, in: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (Eds.), Advances in Neural Information Processing Systems. URL: https://openreview.net/forum?id=_VjQlMeSB_J.

Weng, Y., Zhu, M., Xia, F., Li, B., He, S., Liu, S., Sun, B., Liu, K., Zhao, J., 2023. Large language models are better reasoners with self-verification, in: EMNLP (Findings), pp. 2550–2575. URL: https://aclanthology.org/2023.findings-emnlp.167.

Wu, P., Xie, S., 2024. V*: Guided visual search as a core mechanism in multimodal llms, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024, IEEE. pp. 13084–13094. URL: https://doi.org/10.1109/CVPR52733.2024.01243, doi:10.1109/CVPR52733.2024.01243.

Yang, C.H.H., Gu, Y., Liu, Y.C., Ghosh, S., Bulyko, I., Stolcke, A., 2023. Generative speech recognition error correction with large language models and task-activating prompting, in: 2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), IEEE. pp. 1–8.

Yang, Y.T.C., Newby, T.J., Bill, R.L., 2005. Using socratic questioning to promote critical thinking skills through asynchronous discussion forums in distance learning environments. The american journal of distance education 19, 163–181.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T.L., Cao, Y., Narasimhan, K.R., 2023. Tree of thoughts: Deliberate problem solving with large language models, in: Thirty-seventh Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=5Xc1ecxO1h.

Yu, Q., Zhang, Z., Zhu, R., Yuan, Y., Zuo, X., Yue, Y., Dai, W., Fan, T., Liu, G., Liu, L., Liu, X., Lin, H., Lin, Z., Ma, B., Sheng, G., Tong, Y., Zhang, C., Zhang, M., Zhang, W., Zhu, H., Zhu, J., Chen, J., Chen, J., Wang, C., Yu, H., Song, Y., Wei, X., Zhou, H., Liu, J., Ma, W.Y., Zhang, Y.Q., Yan, L., Qiao, M., Wu, Y., Wang, M., 2025. DAPO: An open-source LLM reinforcement learning system at scale, in: Advances in Neural Information Processing Systems 38 (NeurIPS 2025). URL: https://proceedings.neurips.cc/paper_files/paper/2025/hash/a4277440d50f1f15d2cb4c14f7e0c0d2-Abstract-Conference.html.

Yuan, L., Li, W., Chen, H., Cui, G., Ding, N., Zhang, K., Zhou, B., Liu, Z., Peng, H., 2025. Free process rewards without process labels, in: Forty-second International Conference on Machine Learning. URL: https://openreview.net/forum?id=8ThnPFhGm8.

Yuan, W., Pang, R.Y., Cho, K., Li, X., Sukhbaatar, S., Xu, J., Weston, J.E., 2024. Self-rewarding language models, in: Forty-first International Conference on Machine Learning. URL: https://openreview.net/forum?id=0NphYCmgua.

Yue, Y., Chen, Z., Lu, R., Zhao, A., Wang, Z., Song, S., Huang, G., 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems.

Zhang, F., Xu, J., Wang, C., Cui, C., Liu, Y., An, B., 2025a. Incentivizing LLMs to self-verify their answers, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=MBDWO29Qq6.

Zhang, S., Liu, X., Zhang, X., Liu, J., Luo, Z., Huang, S., Gong, Y., 2025b. Process-based self-rewarding language models, in: Che, W., Nabende, J., Shutova, E., Pilehvar, M.T. (Eds.), Findings of the Association for Computational Linguistics: ACL 2025, Association for Computational Linguistics, Vienna, Austria. pp. 18097–18110. URL: https://aclanthology.org/2025.findings-acl.930/, doi:10.18653/v1/2025.findings-acl.930.

Zhao, J., Fang, K., Cheng, L., 2026. When and what to ask: AskBench and rubric-guided RLVR for LLM clarification, in: Liakata, M., Moreira, V.P., Zhang, J., Jurgens, D. (Eds.), Findings of the Association for Computational Linguistics: ACL 2026, Association for Computational Linguistics, San Diego, California, United States. pp. 17120–17140. URL: https://aclanthology.org/2026.findings-acl.845/, doi:10.18653/v1/2026.findings-acl.845.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Cui, C., Bousquet, O., Le, Q.V., Chi, E.H., 2023. Least-to-most prompting enables complex reasoning in large language models, in: The Eleventh International Conference on Learning Representations. URL: https://openreview.net/forum?id=WZH7099tgfM.

Zuo, Y., Zhang, K., Sheng, L., Qu, S., Cui, G., Zhu, X., Li, H., Zhang, Y., Long, X., Hua, E., Qi, B., Sun, Y., Ma, Z., Yuan, L., Ding, N., Zhou, B., 2025. TTRL: Test-time reinforcement learning, in: The Thirty-ninth Annual Conference on Neural Information Processing Systems. URL: https://openreview.net/forum?id=VuVhgEiu20.

A Guide to the Appendix

1. Sec. B discusses the limitations of AskR and several directions for future work.
2. Sec. C analyzes the token–accuracy trade-off of AskR against PRM-guided and other test-time methods.
3. Sec. D describes the training datasets and the public evaluation benchmarks considered in this work.
4. Sec. E provides implementation details, including prompting templates, training settings, and inference procedures for AskR, the baselines, and the test-time optimizers.
5. Sec. F reports additional experimental analyses, including generalization across reasoner scales, an ablation on the AskR training objective, an analysis of scaling the PRM training budget and main results with standard deviations.
6. Sec. G provides AskR examples for text and vision-language reasoning.

B Limitations and Future Work

AskR has several limitations that suggest directions for future work. First, training AskR still requires verifiable final-answer supervision to align the elicited intrinsic reward with task success. Our experiments therefore focus on reasoning tasks with reliable outcome verification; extending AskR to open-ended or subjective tasks will require alternative forms of outcome-level supervision.

Second, although AskR transfers from its training corpora to multiple held-out benchmarks and remains effective across model families and reasoner scales, we have not established robustness under substantially different domains, reasoning formats, or heterogeneous reasoners without retraining the asker. Broader cross-domain transfer therefore remains an important direction for future work.

Third, AskR requires an additional learned questioning component and introduces extra inference cost from sub-question generation, SubQA reasoning, self-verification, and test-time optimization.

Table 5: **Token–accuracy efficiency on MMLU-Pro.** Input and output denote per-sample token counts, and Total Tokens is their sum. ΔAcc. and ΔTokens are measured relative to CoT. Gain / 1K Extra Tokens measures the accuracy improvement obtained per additional 1K inference tokens. The metric is omitted for L2M because it uses fewer tokens than CoT.

Method	Acc.	Input	Output	Total Tokens	ΔAcc.	ΔTokens	Gain / 1K Extra Tokens
CoT	55.75	291	1,147	1,438	–	–	–
RLVR	**64.19**	291	4,084	4,375	+8.44	+2,937	**2.87**
L2M	56.70	300	840	1,140	+0.95	-298	–
Self-Refine	57.21	684	2,085	2,769	+1.46	+1,331	1.10
ToT	54.00	338	2,265	2,603	-1.75	+1,165	-1.50
VM + VGS	55.36	4,050	2,429	6,479	-0.39	+5,041	-0.08
VM + PF	61.03	4,214	2,768	6,982	+5.28	+5,544	0.95
PRM + BoN	58.40	291	6,147	6,438	+2.65	+5,000	0.53
AskR + BoN	**61.23**	1,083	5,087	**6,170**	**+5.48**	**+4,732**	**1.16**
PRM + TPO	56.18	1,494	16,312	17,806	+0.43	+16,368	0.03
AskR + TPO	**60.61**	2,708	29,892	32,600	**+4.86**	+31,162	**0.16**
PRM + TMPC	60.30	1,282	11,543	12,825	+4.55	+11,387	0.40
AskR + TMPC	**63.76**	1,547	15,302	16,849	**+8.01**	+15,411	**0.52**

Future work could reduce this overhead through adaptive question budgets, early stopping, or more efficient reuse of intermediate computations.

Finally, our current implementation uses binary SubQA-level self-verification and uniform aggregation into a trajectory-level reward. Exploring richer verification signals or adaptive aggregation mechanisms may further improve reward quality while preserving AskR's key advantage of avoiding a separately trained reward model.

C How Token-Efficient Is AskR at Test Time?

AskR introduces additional stages for sub-question generation and self-verification, which increase the sequential inference overhead.

We therefore provide a token-based measure that reflects how effectively additional inference computation translates into task performance. Using CoT as the reference point, we define the marginal token efficiency of a method m as

$$\mathrm{Eff}_{\mathrm{token}}(m) = \frac{\mathrm{Acc}(m) - \mathrm{Acc}(\mathrm{CoT})}{\mathrm{Tokens}(m) - \mathrm{Tokens}(\mathrm{CoT})} \times 1000, \tag{9}$$

where higher values indicate a larger accuracy improvement per additional 1K inference tokens. We report this analysis on MMLU-Pro in Table 5.

While AskR + TMPC achieves the strongest overall performance, it comes with a higher test-time cost due to its more intensive iterative optimization. When balancing accuracy and efficiency, AskR + BoN provides a more favorable operating point, retaining the benefit of the learned intrinsic reward with a substantially lighter test-time procedure.

Notably, despite using the simpler BoN optimizer, AskR + BoN outperforms all test-time baselines on average across the six benchmarks. These results show that AskR does not rely on expensive test-time optimization to be effective: TMPC can be used when maximizing accuracy is the priority, while BoN provides a strong alternative when inference efficiency also matters.

D Training Dataset and Benchmark details

Training datasets. We use different open reasoning corpora for training the language-only and vision-language variants of AskR. For the language-only setting, AskR is trained on 5K verifiable

math and STEM examples drawn from SYNTHETIC-1 (Mattern et al., 2025), an open collection of verified reasoning traces distilled from DeepSeek-R1.

For the vision-language setting, AskR is trained on 5K examples from LONGPERCEPTUALTHOUGHTS (LPT) (Liao et al., 2025), a synthetic dataset of long reasoning traces designed for perceptual and vision-centric tasks. In both cases, AskR is trained without using the provided reasoning traces as supervision; supervision is provided only through final-answer correctness under the reward-alignment objective.

Benchmarks. Our evaluation suite is organized into two parts: text benchmarks and vision-language benchmarks.

Text benchmarks. We evaluate AskR on three text benchmarks in total.

1. **MMLU-Pro** (Wang et al., 2024b) is a more challenging extension of MMLU designed to reduce saturation and emphasize more reasoning-intensive multi-domain question answering. We evaluate on full instances from physics, chemistry, computer science, math, biology, health and engineering subjects.
2. **LAB-Bench** (Laurent et al., 2024) evaluates research-oriented biology capabilities. We drop tasks that require visual inputs or external table/paper extraction, therefore dropping DbQA, FigQA, LitQA2, SuppQA, and TableQA, retaining CloningScenarios, ProtocolQA, and SeqQA.
3. **SciEval** (Sun et al., 2024) is a multi-level benchmark for scientific research ability, covering scientific understanding, analysis, and reasoning across disciplines.

Vision-language benchmarks. We use three core vision-language benchmarks throughout the main paper:

1. **V* Bench** (Wu and Xie, 2024) targets visually demanding reasoning problems that require models to localize and reason over small but task-critical visual evidence, often in high-resolution or visually cluttered images.
2. **MMStar** (Chen et al., 2024b) is a carefully curated vision-language benchmark built from human-selected, vision-indispensable samples, designed to reduce shortcuts from language priors alone. We select categories from coarse perception, fine-grained perception and instance reasoning.
3. **CV-Bench** (Tong et al., 2024) is a vision-centric benchmark introduced with Cambrian-1, covering a broad range of perceptual and visually grounded reasoning abilities.

E Implementation Details

E.1 Prompting-Based Baselines

E.1.1 Chain-of-Thought Reasoning

For Chain-of-Thought reasoning, we prompt the model to solve the problem step by step and provide a clear reasoning process before giving the final answer.

```
SYSTEM_PROMPT =
You are a helpful assistant that solves problems step-by-step.
When given a problem, think through it carefully and show your
reasoning process. Provide a clear, structured solution.
USER_PROMPT = Solve the following problem step by step.
Show your reasoning process clearly.
Problem: {question}
Please provide:
1. Your step-by-step reasoning
2. The final answer in the format: Answer: [Your final answer]
You must use <think> and </think> tags to show your reasoning before
the final answer.
Reasoning: [Your step-by-step thought process]
Answer: [Your final answer]
```

E.1.2 Tree-of-Thought Reasoning

For Tree-of-Thought reasoning, we prompt the model to explore multiple candidate solution paths, evaluate them, and then select the most reliable one. We used $n_branch = 4$ in all evaluations.

```
SYSTEM_PROMPT =
You are a helpful assistant that solves problems using tree-of-thought
reasoning. Explore multiple solution approaches, evaluate each for
correctness, and select the best one.
USER_PROMPT = Solve the following problem using Tree-of-Thought
reasoning.
Problem: {question}
You must use <think> and </think> tags to show your reasoning before
the final answer.
1. Brainstorm {n_branches} distinct approaches:
Think of {n_branches} different methods or strategies to solve this
problem (e.g., algebraic, geometric, proof by contradiction, etc.).
2. Develop each approach:
Follow each approach through, showing your step-by-step work for each
branch.
3. Evaluate each branch:
For each branch, check the arithmetic, verify intermediate results,
and assess whether the reasoning is sound and complete.
4. Select the best branch:
Choose the branch with the most correct and complete reasoning.
Answer: [Your final answer]
```

E.1.3 Least-to-Most Prompting

For Least-to-Most (L2M) prompting (Zhou et al., 2023), we follow the original two-stage procedure. In the *decomposition* stage, the model is prompted to produce an ordered list of at most $K_{\max} = 5$ strictly simpler subproblems, where the last subproblem is required to resolve the original problem. In the *sequential solution* stage, the subproblems are answered one at a time; each step is conditioned on the original problem and all previously solved subproblem–answer pairs, and only the final step is instructed to emit the final answer.

The final answer is extracted from the response to the last subproblem. Per question, this costs one decomposition generation plus one generation per subproblem, which is why L2M uses fewer total tokens than CoT in Table 5. Unlike AskR, the decomposer is the frozen backbone itself, no self-verification is performed, and no reward signal is computed.

```
DECOMPOSE_SYSTEM = You are a careful problem decomposer. Break a
complex problem into a minimal ordered list of strictly simpler
subproblems. The last subproblem must, once answered, yield the final
answer to the original problem. Each subproblem should be answerable
using the answers to the prior subproblems.
DECOMPOSE_USER = Problem:
{question}
{options_block}
Produce an ordered list of at most {max_steps} subproblems. Use this
exact format (one per line, nothing else):
SubQ1: <first subproblem>
SubQ2: <second subproblem>
...
Subproblems should be strictly simpler than the original and build on
each other. The last one should resolve the original problem.
```

```
SOLVE_STEP_SYSTEM = You are answering one subproblem of a decomposed
problem. Use the previous subproblem-answer pairs as context. Be
concise -- answer the current subproblem only. Do not re-state the
entire chain.
SOLVE_STEP_USER = Original problem:
{question}
{options_block}
Subproblems and answers so far:
{prior_subq_suba_pairs}
Current subproblem:
{current_subq}
Answer the current subproblem. {final_answer_hint}
```

`{final_answer_hint}` is empty for all but the last subproblem, where it instructs the model to end with `Final Answer: [A single letter]` for multiple-choice benchmarks or `Final Answer: \boxed{<answer>}` for free-form answers.

E.1.4 Self-Refine

For Self-Refine (Madaan et al., 2023), the model first produces an initial answer using the Chain-of-Thought prompt in Section E.1.1. It then alternates between two passes for K iterations: a *feedback* pass, in which the model critiques its own current response without rewriting it, and a *refine* pass, in which the model rewrites the response given the original problem, its previous response, and the critique. The response after the final refine pass is used as the answer. For vision-language benchmarks, the image is attached to all three passes so that the critic can see the visual input. Per question, this costs $1 + 2K$ generations per trial. As with the other prompting baselines, Self-Refine uses only the frozen backbone and involves no external reward model, self-verification score, or training.

```
FEEDBACK_SYSTEM = You are an expert evaluator. Critique the response
to the given problem. Identify concrete weaknesses: logical errors,
arithmetic mistakes, missing steps, or unsupported claims. Be specific
and constructive. If the answer looks correct, say so and explain why.
FEEDBACK_USER = Problem:
{question}
{options_block}
Response:
{response}
Provide a focused critique of the response. List specific issues,
if any. Do not write a corrected answer here -- just the critique.
```

```
REFINE_SYSTEM = You are a careful reasoner. Given a problem, your
previous response, and a critique of it, write an improved response.
Apply the feedback where appropriate; otherwise stand by your original
reasoning. End with the final answer in the format requested by the
problem.
REFINE_USER = Problem:
{question}
{options_block}
Previous response:
{response}
Critique of previous response:
{feedback}
Write the improved response below. Show your reasoning, then give the
final answer.
{final_answer_hint}
```

E.2 Process Reward Models and Value Models

The prompts for reasoners with PRM and VM are the same as those used for Chain-of-Thought prompting. For the language-only baselines, we train the PRM and VM on the same 5K SYNTHETIC-1 (Mattern et al., 2025) problems used to train AskR. For the vision-language baselines, we analogously use the same 5K LONGPERCEPTUALTHOUGHTS (Liao et al., 2025) examples used for the vision-language AskR.

Rollout collection and step-level labeling. For each training problem x with verifiable answer y, we sample $M = 16$ complete CoT solutions from the frozen reasoner using the CoT prompt with the inference-time decoding settings (temperature 0.8, top-p 0.95). Each solution is segmented into intermediate states $s_1, \ldots, s_T$ at 256-token boundaries, matching the block size used by VGS and PF at inference. To label each intermediate state without human annotation, we follow the Monte Carlo estimation: from every prefix $s_{\le t}$ we sample $M' = 8$ independent continuations from the frozen reasoner, run each to completion, and check its final answer with the task verifier. This yields, for every prefix, an empirical success rate

$$\hat{V}(s_{\le t}) = \frac{1}{M'} \sum_{j=1}^{M'} V\big(\hat{y}_j,\, y\big) \in [0, 1],$$

which measures how often the reasoner reaches the correct answer when continuing from that prefix. Labels are therefore grounded in the reasoner's own downstream behavior rather than inherited from a single trajectory's terminal outcome, which would conflate early correct steps with later errors.

Value model (VM). The VM regresses $\hat{V}(s_{\le t})$ directly: given a prefix, the scalar head (passed through a sigmoid) is trained with a mean-squared-error loss against the empirical success rate. The VM thus estimates the expected final correctness of a partial solution, which is exactly the quantity VGS and PF use to rank partial trajectories.

Process reward model (PRM). The PRM instead classifies whether each individual step is correct. We binarize the same Monte Carlo estimates with the hard-estimation rule of Math-Shepherd: step s_t is labeled correct ($z_t = 1$) if $\hat{V}(s_{\le t}) > 0$, i.e., at least one continuation from that prefix reaches the correct answer, and incorrect ($z_t = 0$) otherwise. Because a prefix that has already gone wrong cannot recover, once a step is labeled incorrect all subsequent steps in that solution are labeled incorrect as well. The PRM is trained with a binary cross-entropy loss on z_t at every step position.

Labeling cost. This labeling procedure is substantially more expensive than training AskR: per training problem it requires M full solutions plus M' continuations from every intermediate prefix, i.e., $M \times \bar{T} \times M'$ additional rollouts where $\bar{T}$ is the mean number of steps, compared with the 8 rollouts per problem used to train AskR with GRPO. We report this to make clear that the PRM/VM baselines are not under-resourced: they receive strictly more supervision signal per problem than AskR, and Appendix F.3 further shows that giving the PRM six times more problems does not close the gap.

In the text-only setting, both PRM and VM are initialized from `Qwen2.5-7B-Instruct`. In the vision-language setting, we use `Qwen2.5-VL-7B-Instruct` as the backbone. All models are fine-tuned for 3 epochs with a per-device batch size of 4 and 8 gradient accumulation steps with AdamW optimizer, resulting in an effective batch size of 32. We set the learning rate to 2×10^{-5} and use a warmup ratio of 0.1.

E.3 AskR Prompting

E.3.1 Prompts for sub-questions generation

```
SYSTEM_PROMPT = You are a helpful assistant that breaks down complex
problems into simpler sub-questions.
When given a problem, decompose it into a chain of sub-questions that,
when answered sequentially, will lead to the final answer.
USER_PROMPT = Break down the following problem into a chain of
sub-questions. Each sub-question should:
1. Be simpler than the original problem
2. Build upon previous sub-questions
3. Lead step-by-step toward the final answer
4. Be answerable independently
Problem: {question}
Please generate N sub-questions in the following format,
you should decide how many N by yourself (But N <= {max_subqs_len}):
SubQ1: [First sub-question]
SubQ2: [Second sub-question]
SubQ3: [Third sub-question]
SubQ4: [Fourth sub-question]
SubQ5: [Fifth sub-question] ...
Only output the sub-questions, one per line, in the format
"SubQN: [question]".
```

E.3.2 Prompts for sub-answers generation

```
SYSTEM_PROMPT = You are a helpful assistant that solves complex
problems by breaking them down into sub-questions.
Answer each sub-question clearly and concisely, then use all the
sub-answers to provide the final answer.
USER_PROMPT = Original Problem: {question}
To solve this problem, answer the following sub-questions one by one.
Each answer should be clear and concise.
Sub-Questions:
{subq_list_previously_generated}
Please provide your response in the following format:
SubA1: [Your answer to the first sub-question]
SubA2: [Your answer to the second sub-question]
SubA3: [Your answer to the third sub-question]
... (continue for all sub-questions)
After answering all sub-questions,
synthesize the information to provide the final answer:
Answer: [Your final answer]
```

E.3.3 Prompts for intrinsic rewards generation

```
SYSTEM_PROMPT = You are an expert at verifying the correctness of
reasoning chains. You will evaluate all sub-questions and sub-answers
simultaneously to assess their collective quality.
USER_PROMPT = Original Problem: {original_question}
Reasoning Chain (All SubQ-SubA pairs):
{subqa_text}
Task: Evaluate the correctness of EACH sub-answer in the context of
the entire reasoning chain.
For each SubQ-SubA pair, assess:
1. Relevance: Does the sub-answer directly address sub-question?
2. Correctness: Is the sub-answer factually accurate?
3. Completeness: Does the sub-answer provide sufficient information?
4. Consistency: Is the sub-answer consistent with other sub-answers
in the chain?
Provide your analysis in the following format:
For each SubQ-SubA pair (1, 2, 3, ...), analyze its correctness
considering the entire chain.
VERDICTS:
Provide one verdict per line in the exact format:
SubQ 1: [true/false]
SubQ 2: [true/false]
SubQ 3: [true/false]
...
Where "true" means the sub-answer is correct and "false" means it is
incorrect or problematic.
```

E.3.4 Training Details

We adopt `Qwen2.5-7B-Instruct` and `Qwen2.5-VL-7B-Instruct` as the backbone models for the question generator in the language-only and vision-language settings, respectively. For the model-family and model-scale experiments in Section 5.4, the question generator is instead initialized from `Qwen3-4B` or `Gemma3-4B`. In the scale experiments, the `Qwen3-4B` AskR is trained once and reused unchanged across all reasoner sizes. Training is conducted on a single NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition GPU, and the implementation is based on the Unsloth library (Han and Han, 2023).

The GRPO-based question generator is optimized on 5K verifiable problems: math and STEM problems from SYNTHETIC-1 (Mattern et al., 2025) for the language-only setting, and examples from LONGPERCEPTUALTHOUGHTS (Liao et al., 2025) for the vision-language setting. We use 8 rollouts per prompt for GRPO training. Regardless of backbone or modality, all AskR models are trained with the same hyperparameters: 2 epochs, a per-device batch size of 4 with 8 gradient accumulation steps (effective batch size 32), the AdamW optimizer, FP16 precision, a learning rate of 2×10^{-5}, a warmup ratio of 0.1, KL regularization coefficient $\beta = 0.0$, token-level clipped surrogate of DAPO (Yu et al., 2025) and gradient clipping with a maximum gradient norm of 1.0. We train with LoRA using rank $r = 32$ and scaling factor $\alpha = 64$. For sub-question generation, the question generator is constrained to produce at most 10 sub-questions.

E.4 Inference Details for Different Test-Time Optimizers

All methods are evaluated with 8 independent trials per problem (i.e., 8 final answers generated), using sampling temperature=0.8 and top-p=0.95. The final result score is the average of 8 trials.

E.4.1 Best-of-N (BoN)

BoN is the simplest optimizer we consider and involves no iterative refinement. For each trial, $N = 8$ candidate responses are sampled independently from the reasoner, each is scored by the reward, and the highest-scoring candidate is returned as the final answer.

BoN-AskR. AskR first generates a SubQ sequence $\mathcal{Q}$ for the problem x; the reasoner π_θ then produces N candidate SubQA trajectories conditioned on $\mathcal{Q}$, each scored by the intrinsic reward $r_\phi(x, \tau)$ in Eq. 4. The candidate with the highest r_ϕ is selected.

BoN-CoT-RM. N chain-of-thought responses are sampled with the CoT prompt in Section E.1.1 and scored by the externally trained PRM; the highest-scoring response is selected. This variant replaces AskR's intrinsic reward with an external reward model under the same selection rule.

E.4.2 Textual Model Predictive Control (TMPC)

We evaluate two TMPC variants that iteratively refine reasoning over $D = 3$ iterations, each generating $N = 8$ candidate responses per iteration.

TMPC-AskR. At each iteration, AskR g_ϕ produces a fixed set of sub-questions $\mathcal{Q}$ for the input problem x. The reasoner π_θ then generates N candidate SubQA trajectories conditioned on $\mathcal{Q}$, each scored by the intrinsic reward $r_\phi(x, \tau)$ defined in Eq. 4. We define a candidate as *high-quality* if it ranks among the top-3 candidates by r_ϕ; candidates failing either condition are discarded. Each high-quality trajectory is parsed into its constituent sub-question–answer pairs, which are stored in a SEGMENTEDBUFFER partitioned into opening, middle, and ending slots according to their position in the trajectory. The buffer accumulates across iterations, so segments selected in earlier iterations remain available alongside newly added ones. Its content is injected as in-context guidance into the next iteration's prompt, steering subsequent generations toward SubQA patterns that received high intrinsic reward. After D iterations, a final rollout produces the answer. Throughout, r_ϕ is the only signal used for selection; no ground-truth labels are involved at inference time.

TMPC-CoT-RM. This variant follows the same iterative structure but operates on chain-of-thought responses rather than SubQA trajectories. At each iteration, N CoT responses are generated and scored by the externally trained PRM; responses that rank in the top-3 under the PRM are treated as high-quality, and their reasoning steps are stored and injected via the same accumulating SEGMENTEDBUFFER. This baseline thus replaces AskR's intrinsic reward with an external reward model while keeping the optimizer identical.

E.4.3 Test-Time Preference Optimization (TPO)

TPO aligns model outputs with a reward model at inference time without updating model weights. Instead of gradient descent on parameters, TPO performs optimization *in textual form*: the model iteratively critiques and revises its own responses guided by reward signals.

Each TPO round proceeds as follows. First, N candidate responses are sampled and scored by the reward model, identifying a chosen and a rejected response. Then, for D revision rounds: **(i)** the model critiques the chosen vs. rejected pair to produce a textual loss; **(ii)** converts the critique into improvement instructions (textual gradient); **(iii)** generates N new candidates guided by these instructions; and **(iv)** rescores and updates the chosen/rejected pair. The highest-scoring response across all iterations is returned.

We run 8 independent TPO trials per question. With $N = 8$ candidate per iteration and $D = 3$ revision rounds.

TPO-AskR. Candidate responses are structured QA trajectories, scored by the intrinsic reward $r_\phi(x, \tau)$ in Eq. 4. The textual critique and gradient steps operate over sub-question–answer pairs rather than free-form CoT.

TPO-CoT-RM. Candidate responses are chain-of-thought responses scored by an external reward model, serving as a baseline that replaces AskR's intrinsic reward with an external reward model.

E.4.4 Value-Guided Search (VGS)

VGS (Wang et al., 2025a) performs beam search at the block level (e.g., 256-token), guided by a value model. At each step, every active beam is expanded into $W = 8$ candidate continuations, scored by the value model, and the top-$B = 3$ beams are retained. The total generation budget per

search step is $B \times W = 24$ candidates. The final answer is selected via Weighted Majority Voting (WMV) over completed beams, using value model scores as weights.

We evaluate the CoT with value model, where candidate continuations are chain-of-thought reasoning blocks scored by an externally trained value model.

E.4.5 Particle Filtering (PF)

PF (Puri et al., 2025) maintains $N = 8$ particles (partial reasoning trajectories) and refines them via Sequential Monte Carlo sampling. At each block-level step (e.g., 256 tokens), each particle generates one block of tokens, all particles are scored by a value model, and particles are resampled via softmax-weighted multinomial sampling. We evaluate CoT with PF using an externally trained value model.

E.5 Training Details of RLVR Fine-Tuning

Following the training setup of AskR, we use `Qwen2.5-7B-Instruct` and `Qwen2.5-VL-7B-Instruct` as the backbone models for the RLVR baseline in the language-only and vision-language settings, respectively. Training is conducted on a single NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition GPU, and the implementation is based on the Unsloth library (Han and Han, 2023).

For the Qwen2.5 backbones, RLVR training proceeds in two stages. We first perform supervised fine-tuning (SFT) on reasoning traces: 100K math and STEM long chain-of-thought traces from SYNTHETIC-1 (Mattern et al., 2025), constructed by DeepSeek-R1 (Guo et al., 2025), for the language-only model, and 25K reasoning traces from LONGPERCEPTUALTHOUGHTS (Liao et al., 2025) for the vision-language model. We then further optimize each model with GRPO on 5K verifiable problems drawn from the same source as its SFT data (SYNTHETIC-1 for language-only, LONGPERCEPTUALTHOUGHTS for vision-language). Note that the GRPO training set is identical to the one used to train AskR in the corresponding setting, so RLVR and AskR see the same verifiable problems; RLVR additionally consumes the SFT traces, which AskR does not use.

For the model-family experiments in Section 5.4, the RLVR baselines are trained from `Qwen3-4B` and `Gemma3-4B` using the GRPO stage only, without the preceding SFT stage, since both backbones are already released with reasoning capabilities; the GRPO data and hyperparameters are otherwise identical to those above.

Both the text-only and vision-language models are trained with the same hyperparameters. SFT runs for 3 epochs in BF16; GRPO runs for 2 epochs in FP16 with 8 rollouts per prompt. Both stages use a per-device batch size of 4 with 8 gradient accumulation steps (effective batch size 32), the AdamW optimizer, a learning rate of 2×10^{-5}, a warmup ratio of 0.1, and gradient clipping with a maximum gradient norm of 1.0. The KL regularization coefficient β in GRPO is set to 0.0.

F Additional Results

F.1 Generalization Across Reasoner Scales

To examine whether AskR generalizes across backbone scales, we fix the AskR to `Qwen3-4B` and use it to steer `Qwen3` reasoners ranging from 1.7B to 32B under TMPC. CoT with the same reasoner serves as the reference, and we additionally report the same AskR before training, using the identical SubQA interface, to isolate the contribution of training. Results are reported in Table 6. Trained AskR + TMPC consistently improves over CoT on every benchmark and at every reasoner scale, including reasoners substantially larger than the fixed 4B AskR. The average improvement decreases monotonically as the reasoner becomes stronger, from **+9.96** points at 1.7B to **+4.70** points at 32B, while remaining positive throughout. In contrast, AskR before training provides only marginal and inconsistent gains over CoT (+0.43 to +1.69 points on average, and below CoT on some individual benchmarks), and trained AskR outperforms it by +3.79 to +8.27 points at every scale. These results show that the trained AskR is not tied to the scale of the reasoner used during training, and that the benefit stems from the learned questioning policy rather than from the SubQA interface itself: a single 4B AskR can be reused to guide a family of reasoners from 1.7B to 32B, with diminishing additional gains as the underlying reasoner becomes stronger.

Table 6: **Generalization across reasoner scales.** A single AskR initialized from Qwen3-4B is reused to guide Qwen3 reasoners ranging from 1.7B to 32B under TMPC. CoT with the same reasoner serves as the reference, and the before-training variant uses the same SubQA interface. AskR after training + TMPC improves over both on every benchmark and at every reasoner scale.

Reasoner	Method	MMLU-Pro	LAB-Bench	SciEval	Avg.
Qwen3-1.7B	CoT	45.76	19.00	68.50	44.42
	AskR (Before Training) + TMPC	47.14	21.75	69.43	46.11
	AskR (After Training) + TMPC	**57.43**	**33.33**	**72.38**	**54.38**
Qwen3-4B	CoT	69.60	34.98	71.60	58.73
	AskR (Before Training) + TMPC	73.57	35.46	72.14	60.39
	AskR (After Training) + TMPC	**78.42**	**44.83**	**76.46**	**66.57**
Qwen3-8B	CoT	75.11	36.82	72.14	61.36
	AskR (Before Training) + TMPC	76.29	38.72	73.43	62.81
	AskR (After Training) + TMPC	**80.29**	**44.33**	**78.29**	**67.64**
Qwen3-14B	CoT	77.43	43.03	74.43	64.96
	AskR (Before Training) + TMPC	79.57	45.33	74.00	66.30
	AskR (After Training) + TMPC	**82.86**	**47.27**	**80.14**	**70.09**
Qwen3-32B	CoT	80.62	45.98	77.84	68.14
	AskR (Before Training) + TMPC	80.29	45.71	79.71	68.57
	AskR (After Training) + TMPC	**84.70**	**48.85**	**84.97**	**72.84**

Table 7: Per-benchmark reward–correctness alignment before and after AskR training. Metrics are computed between the trajectory-level intrinsic reward and binary final-answer correctness on all evaluation examples used for each benchmark.

Backbone / Stage	Metric	MMLU-Pro	LAB-Bench	SciEval
Qwen2.5-7B / Before	Pearson	0.1214	0.1413	0.1823
	Spearman	0.0213	0.0922	0.1559
	AUROC	0.5101	0.5542	0.5791
Qwen2.5-7B / After	Pearson	0.4731	0.3832	0.4512
	Spearman	0.4310	0.2898	0.4244
	AUROC	0.7601	0.6142	0.7591
Qwen3-4B / Before	Pearson	0.1053	0.2374	0.1392
	Spearman	0.1201	0.2487	0.0914
	AUROC	0.5742	0.6232	0.5901
Qwen3-4B / After	Pearson	0.5531	0.4218	0.5513
	Spearman	0.5013	0.4074	0.4381
	AUROC	0.7961	0.6401	0.7848

F.2 Ablation on the AskR Training Objective: Intrinsic Reward Alignment vs. Direct Task Optimization

Per-benchmark reward–correctness alignment. To complement the macro-averaged results in Figure 4, Table 7 reports reward–correctness alignment separately on the full evaluation sets of MMLU-Pro, LAB-Bench, and SciEval, both before and after AskR training. We compute Pearson correlation, Spearman correlation, and AUROC between the trajectory-level intrinsic reward $r_\phi(x, \tau)$ and binary final-answer correctness $\mathcal{V}(\hat{y}, y)$. AskR training improves all three metrics across all three benchmarks for both Qwen2.5-7B and Qwen3-4B, showing that the improvement in reward alignment is consistent across benchmarks rather than driven by a single dataset.

Correctness-aligned reward vs. direct task optimization. AskR is trained with the correctness-aligned objective in Eq. 5, which couples the intrinsic reward r_ϕ with the signed final-answer outcome. This objective is fundamentally different from directly training AskR to maximize final-answer correctness $V(\hat{y}, y)$. The latter rewards questions whenever they help the reasoner solve the original task, and therefore encourages AskR to learn useful decompositions or sub-questions that

Table 8: **Ablation on the training reward.** AskR trained with the correctness-aligned reward (Eq. 5) versus answer correctness alone (the RLVR objective applied to the AskR).

Backbone	Training reward	MMLU-Pro	LAB-Bench	SciEval	Avg.
Qwen2.5-7B	Correctness-aligned (Eq. 5)	**63.76**	**37.29**	**70.45**	**57.17**
	Answer correctness only	59.08	33.47	68.53	53.69
Qwen3-4B	Correctness-aligned (Eq. 5)	**78.42**	**44.83**	**76.46**	**66.57**
	Answer correctness only	71.98	40.03	73.54	61.85

Table 9: **Effect of PRM training budget.** All rows use TMPC as the test-time optimizer. PRM-5K and AskR-5K are the TMPC rows of Table 2, trained on matched 5K budgets. PRM-30K increases the PRM training set to 30K examples (six times the AskR budget) while AskR remains at 5K.

	Language-only				Vision-language			
Reward	MMLU-Pro	LAB-Bench	SciEval	**Avg.**	V* Bench	MMStar	CV-Bench	**Avg.**
PRM-5K	60.30	31.17	47.86	46.44	75.78	58.40	66.98	67.05
PRM-30K	60.23	28.49	51.95	46.89	77.80	57.31	68.64	67.91
AskR-5K	**63.76**	**37.29**	**70.45**	**57.17**	**84.37**	**63.23**	**76.43**	**74.68**

directly make the problem easier. In contrast, our objective does not optimize the questions for task solving alone. Instead, it trains AskR to induce reasoning states for which self-verification is *aligned with* final-answer correctness: high intrinsic reward should be associated with successful trajectories, whereas the same signal should be suppressed on unsuccessful ones.

We compare these two objectives on `Qwen2.5-7B` and `Qwen3-4B`, while holding all other factors fixed. Both variants optimize AskR with GRPO under identical hyperparameters, and both are evaluated at test time with TMPC using the same intrinsic reward r_ϕ. Thus, the comparison isolates whether it is sufficient to learn questions that directly improve task solving, or whether the questions must instead be trained to make self-verification a reliable reward signal. Results are reported in Table 8. The correctness-aligned objective outperforms direct final-answer optimization on every benchmark for both backbones. This gap shows that learning to ask helpful questions is not sufficient: questions that make the original problem easier do not necessarily produce self-verification signals that are informative for guiding test-time optimization. Rather, AskR learns a reasoning interface that jointly structures the SubQA trajectory and elicits a self verification signal aligned with final correctness. These two effects are coupled by design rather than independently optimized.

F.3 Scaling the PRM training Budget

Here, we test whether AskR's advantage can be explained by limited PRM training data. We increase the PRM training set from 5K to 30K examples, which is six times the training budget used by AskR, while keeping the AskR training budget fixed at 5K. All variants are evaluated on `Qwen2.5-7B-Instruct` and `Qwen2.5-VL-7B-Instruct` under the same TMPC test-time optimizer. As shown in Table 9, increasing the PRM training budget to 30K improves the average by only +0.65 points. Despite using six times more training examples, PRM-30K remains substantially behind AskR-5K on every benchmark. These results indicate that AskR's advantage over PRM guidance is not a consequence of a restricted PRM training budget; the learned intrinsic reward remains substantially more effective even when the PRM is given considerably more supervision.

F.4 Main Results with Standard Deviations

Table 10 provides an expanded version of Table 2, reporting mean $\pm$ standard deviation across 8 independent inference sampling trials. These standard deviations characterize inference-time sampling variability, rather than variability across independently trained checkpoints.

Table 10: Main results with mean ± standard deviation over 8 independent inference sampling trials per problem. Rows and columns match Table 2.

Method	Language-only				Vision-language			
	MMLU-Pro	LAB-Bench	SciEval	**Avg.**	V* Bench	MMStar	CV-Bench	**Avg.**
Training-based reference								
RLVR	**64.19** ±0.33	33.75 ±1.45	68.78 ±0.23	55.57 ±0.50	79.50 ±1.94	60.93 ±1.01	72.91 ±0.62	71.11 ±0.76
Prompting-based								
CoT	55.75 ±0.22	32.63 ±0.91	48.75 ±0.28	45.71 ±0.33	72.50 ±2.15	59.13 ±1.66	68.47 ±0.83	66.70 ±0.95
L2M	56.70 ±0.31	29.80 ±0.88	43.23 ±0.43	43.24 ±0.34	72.94 ±2.01	55.21 ±1.40	68.96 ±0.92	65.70 ±0.87
ToT	54.00 ±0.30	33.72 ±1.20	43.27 ±0.30	43.66 ±0.42	75.06 ±1.90	60.53 ±1.50	66.25 ±0.80	67.28 ±0.85
Self-Refine	57.21 ±0.41	33.53 ±1.02	59.53 ±0.33	50.09 ±0.38	73.53 ±2.30	57.00 ±1.89	66.34 ±0.90	65.62 ±1.04
Value-model (VM)-guided methods								
VM + VGS	55.36 ±0.21	32.20 ±1.53	48.70 ±0.44	45.42 ±0.54	71.24 ±1.80	61.83 ±1.50	74.60 ±0.75	69.22 ±0.82
VM + PF	61.03 ±0.60	31.89 ±0.96	51.99 ±0.44	48.30 ±0.40	78.14 ±1.70	61.05 ±1.40	71.74 ±0.70	70.31 ±0.77
BoN (matched optimizer)								
PRM + BoN	58.40 ±0.55	30.49 ±1.69	49.67 ±0.40	46.19 ±0.61	72.75 ±1.80	58.80 ±1.70	69.54 ±0.80	67.03 ±0.87
AskR + BoN	61.23 ±0.56	35.05 ±0.55	65.43 ±1.05	53.90 ±0.44	77.48 ±0.47	61.73 ±1.86	72.53 ±0.74	70.58 ±0.69
TPO (matched optimizer)								
PRM + TPO	56.18 ±0.61	29.70 ±1.17	48.50 ±0.44	44.79 ±0.46	78.01 ±1.70	42.66 ±1.60	72.46 ±0.75	64.38 ±0.82
AskR + TPO	60.61 ±0.60	33.19 ±1.20	63.10 ±0.40	52.30 ±0.47	76.43 ±1.80	62.40 ±1.50	71.07 ±0.70	69.97 ±0.82
TMPC (matched optimizer)								
PRM + TMPC	60.30 ±0.55	31.17 ±1.70	47.86 ±0.35	46.44 ±0.61	75.78 ±1.60	58.40 ±1.30	66.98 ±0.65	67.05 ±0.72
AskR + TMPC	63.76 ±0.29	**37.29** ±1.47	**70.45** ±0.30	**57.17** ±0.51	**84.37** ±0.59	**63.23** ±0.86	**76.43** ±0.44	**74.68** ±0.38

G Qualitative Examples of AskR

G.1 Examples for text reasoning

The following example illustrates this shift on a multi-step geometry problem. Before training, AskR asks superficially relevant questions, but the sequence contains repetition and misses the key convexity constraint until too late, leading the trajectory toward an incorrect low-angle solution.

After training, the sub-questions become more structured: AskR first parameterizes the arithmetic progression, derives the governing equation, enforces convexity, solves for the maximal feasible step size, and finally verifies that the resulting angles satisfy all constraints. The learned sequence is therefore not just more detailed; it is better organized around the intermediate checks that determine correctness.

Question: *The degree-measures of the interior angles of convex hexagon TIEBRK are all integers in arithmetic progression. Compute the least possible degree-measure of the smallest interior angle in hexagon TIEBRK.* Ground truth: **65**

AskR Before Training

SubQ1: What is the sum of the interior angles of a convex hexagon?
SubA1: The sum of the interior angles of a convex hexagon is 720 degrees.

SubQ2: In an arithmetic progression, how can the sum of the six interior angles be expressed in terms of the first angle and the common difference?
SubA2: If the angles are in arithmetic progression, we can denote them as

$$a,\ a+d,\ a+2d,\ a+3d,\ a+4d,\ a+5d,$$

where a is the first angle and d is the common difference. Their sum is

$$a+(a+d)+(a+2d)+(a+3d)+(a+4d)+(a+5d)=6a+15d=720.$$

Thus,

$$2a+5d=240.$$

SubQ3: What is the smallest possible value for the common difference if all angles are integers?
SubA3: To minimize a, we need to maximize d. Since

$$d=\frac{240-2a}{5},$$

for d to be an integer, $240-2a$ must be divisible by 5. The smallest possible value for a occurs when $2a=10$, so

$$a=5,\quad d=\frac{240-10}{5}=46.$$

SubQ4: If the smallest angle is minimized, what are the possible values for the angles of the hexagon?
SubA4: If $a=5$ and $d=46$, the angles are

$$5,\ 51,\ 97,\ 143,\ 189,\ 235.$$

Since 235° is not possible in a convex hexagon, we adjust. The next smallest possible value for a is 10, giving

$$10,\ 54,\ 98,\ 142,\ 186,\ 230.$$

Again, this is not valid, so we try $a=15$, which gives

$$15,\ 57,\ 99,\ 141,\ 183,\ 225.$$

All these angles are less than 180°, so the smallest possible degree-measure is 15.

SubQ5: What is the smallest possible degree-measure of the smallest interior angle in hexagon TIEBRK?
SubA5: The smallest possible degree-measure of the smallest interior angle in hexagon TIEBRK is $\boxed{15}$.

AskR After Training

SubQ1: Given that the degree-measures of the interior angles of hexagon TIEBRK are in arithmetic progression, express the angles in terms of the first angle and the common difference.
SubA1: The angles can be expressed as

$$a,\ a+d,\ a+2d,\ a+3d,\ a+4d,\ a+5d,$$

where a is the first angle and d is the common difference.

SubQ2: Write the equation for the sum of the interior angles of a hexagon.
SubA2: The sum of the interior angles of a hexagon is

$$720^\circ.$$

SubQ3: Use the sum of the interior angles of a hexagon to find the relationship between a and d.
SubA3: The sum of the angles is

$$a+(a+d)+(a+2d)+(a+3d)+(a+4d)+(a+5d)=6a+15d=720.$$

This simplifies to

$$2a+5d=240.$$

SubQ4: Solve the equation from SubQ3 to find the relationship between the first angle and the common difference.
SubA4: Rearranging,

$$2a=240-5d \quad \Rightarrow \quad a=120-\frac{5d}{2}.$$

For a to be an integer, d must be even. Let $d=2k$, where k is an integer. Then

$$a=120-5k.$$

SubQ5: What conditions must hold since the hexagon is convex?
SubA5: Since the hexagon is convex, every interior angle must be strictly less than 180°, and the smallest angle must be positive. Thus,

$$a>0 \quad \text{and} \quad a+5d<180.$$

SubQ6: Find the largest valid integer value of k such that all angles remain less than 180°.
SubA6: Since $d=2k$, the angles are

$$a,\ a+2k,\ a+4k,\ a+6k,\ a+8k,\ a+10k.$$

Using $a=120-5k$, the largest angle must satisfy

$$a+10k<180.$$

Substituting,

$$120-5k+10k<180 \quad \Rightarrow \quad 5k<60 \quad \Rightarrow \quad k<12.$$

Also,

$$a=120-5k>0 \quad \Rightarrow \quad k<24.$$

Hence the largest possible integer is $k=11$.

SubQ7: Compute the corresponding values of a and d.
SubA7: With $k=11$,

$$d=2k=22, \qquad a=120-5k=120-55=65.$$

SubQ8: Verify the resulting angles form a valid convex hexagon.
SubA8: The angles are

$$65,\ 87,\ 109,\ 131,\ 153,\ 175.$$

They are all integers, they form an arithmetic progression, and each is less than 180°. Therefore, this is valid.

SubQ9: What is the least possible degree-measure of the smallest interior angle?
SubA9: The least possible degree-measure of the smallest interior angle is $\boxed{65}$.

G.2 Examples for visual reasoning

The next example shows the same effect in vision-language reasoning. Before training, AskR repeatedly asks variants of essentially the same question (whether a tissue box is present or what its

color might be), but fails to establish the crucial perceptual evidence first. Because the questioning is repetitive rather than diagnostic, self-verification has little useful structure to work with. After training, AskR adopts a more effective order: it first inventories salient objects in the scene, then localizes the candidate tissue box, then characterizes its appearance, and only then maps that evidence onto the answer choices. This illustrates the broader pattern we observe in visual reasoning: AskR learns to first identify the relevant region or object, then interpret it, and finally verify the answer against the available options.



Question: *What is the color of the tissue box? (A) gray (B) white (C) black (D) blue*
Ground truth: **D**

AskR Before Training

SubQ1: Is there a tissue box visible in the image?
SubA1: No, there is no tissue box visible in the image. The image shows a white truck with a trailer and some greenery in the background.

SubQ2: If there is a tissue box, what is its color?
SubA2: There is no tissue box in the image, so this question cannot be answered.

SubQ3: Is the color of the tissue box gray?
SubA3: Since there is no tissue box in the image, this question cannot be answered.

SubQ4: Is the color of the tissue box white?
SubA4: Since there is no tissue box in the image, this question cannot be answered.

SubQ5: Is the color of the tissue box black?
SubA5: Since there is no tissue box in the image, this question cannot be answered.

SubQ6: Is the color of the tissue box blue?
SubA6: Since there is no tissue box in the image, this question cannot be answered.
Final Answer: None

AskR After Training

SubQ1: What objects/features are visible in the image?
SubA1: The image shows a white truck with the brand "MAN" on the front. There is a tree in the foreground, some vegetation, and a guardrail in the foreground as well. There is a tissue box visible on the truck's dashboard.

SubQ2: Can you identify any objects that resemble a tissue box in the image?
SubA2: Yes, there is a small, rectangular object on the dashboard of the truck that resembles a tissue box.

SubQ3: If a tissue box is present, what is its color based on the visible features?
SubA3: The tissue box appears to be a light purple or lavender color.

SubQ4: Considering the options provided, which color best matches the tissue box's appearance?
SubA4: The color that best matches the tissue box's appearance is blue.

SubQ5: How does the color of the tissue box compare with the colors of the truck and the surrounding environment?
SubA5: The truck is white, and the surrounding environment includes green vegetation and a gray guardrail. The tissue box is a distinct light purple/lavender color, which is different from the truck and the environment.

SubQ6: Does the presence of any other objects in the image provide any additional clues about the color of the tissue box?
SubA6: No, the presence of other objects in the image does not provide additional clues about the color of the tissue box. The color of the tissue box is distinct and does not match the colors of the truck or the environment.

SubQ7: Considering the context of the image and the options provided, can you justify why one answer choice might be more likely than the others?
SubA7: The color of the tissue box is distinct and does not match the white color of the truck or the green and gray colors of the environment. The closest match to the tissue box's color among the given options is blue.

SubQ8: After carefully analyzing the image and the options, which answer choice do you believe best represents the color of the tissue box?
SubA8: The best answer choice that represents the color of the tissue box is D (blue).

Final Answer: D