RAST: Reasoning Activation in LLMs via Small-model Transfer

Siru Ouyang[1], Xinyu Zhu[2], Zilin Xiao[3], Minhao Jiang[4], Yu Meng[2], Jiawei Han[1]
[1] University of Illinois Urbana-Champaign, [2] University of Virginia
[3] Rice University, [4] GE HealthCare
siruo2@illinois.edu

Abstract

Reinforcement learning (RL) has become a powerful approach for improving the reasoning capabilities of large language models (LLMs), as evidenced by recent successes such as OpenAI's o1 and Deepseek-R1. However, applying RL at scale remains intimidatingly resource-intensive, requiring multiple model copies and extensive GPU workloads. On the other hand, while being powerful, recent studies suggest that RL does not fundamentally endow models with new knowledge; rather, it primarily reshapes the model's output distribution to activate reasoning capabilities latent in the base model. Building on this insight, we hypothesize that the changes in output probabilities induced by RL are largely model-size invariant, opening the door to a more efficient paradigm: training a small model with RL and transferring its induced probability shifts to larger base models. To verify our hypothesis, we conduct a token-level analysis of decoding trajectories and find high alignment in RL-induced output distributions across model scales, validating our hypothesis. Motivated by this, we propose RAST, a simple yet effective method that transfers reasoning behaviors by injecting RL-induced probability adjustments from a small RL-trained model into larger models. Experiments across multiple mathematical reasoning benchmarks show that RAST substantially and consistently enhances the reasoning capabilities of base models while requiring significantly lower GPU memory than direct RL training, sometimes even yielding better performance than the RL-trained counterparts. Our findings offer new insights into the nature of RL-driven reasoning and practical strategies for scaling its benefits without incurring its full computational cost. The project page of RAST is available at https://ozyyshr.github.io/RAST/.

1 Introduction

Reinforcement learning (RL) [25, 62] has emerged as a powerful and prevalent paradigm for enhancing the reasoning capabilities of large language models (LLMs) [16, 51, 70, 56]. Notably, recent successes such as OpenAI's o1 model [23] and Deepseek-R1 [14] have demonstrated substantial improvements through learning from oracle-verified feedback, employing advanced RL algorithms including Proximal Policy Optimization (PPO) [55] and Group Relative Policy Optimization (GRPO) [57]. However, RL is notoriously inefficient and resource-intensive [58] — it requires loading multiple copies of the same-sized models (e.g., policy, critic, reference, reward) with extensive training GPU memory workloads. Additionally, traditional RL algorithms (e.g., PPO) typically require multiple iterations, each involving interdependent stages such as rollout, replay, and optimization [32].

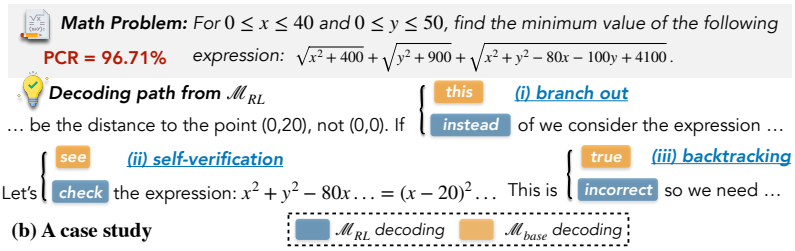
On the other hand, recently, there have been a bunch of works studying behaviors of RL-trained models, aiming to answer the question, *"how does RL elicit reasoning capabilities in LLMs?"* [44, 57,

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

https://doi.org/10.52202/085713-1663

Model Scale	PCR
7B	96.03
14B	95.60
32B	95.22

(a) PCR for different model scales



(b) A case study

Figure 1: *(a) PCR (path coverage rate) across different model scales. (b) A case study* revealing the decoding path of $\mathcal{M}_{\text{base}}$ and its RL-trained version $\mathcal{M}_{\text{RL}}$. Only a very small subset of tokens differ on the decoding path between $\mathcal{M}_{\text{base}}$ and $\mathcal{M}_{\text{RL}}$, which indicates particular reasoning behaviors.

4]. It is increasingly believed that RL does not endow LLMs with fundamentally new knowledge [74, 12]. Instead, it serves to elicit and amplify reasoning behaviors already present within base models [78, 41, 8]. For example, branching out for alternative solution paths, backtracking from incorrect steps [11, 52], and self-verification for generation [63]. The aforementioned studies lead to a major hypothesis [34] in our paper as follows:

> **Hypothesis:** RL activates latent reasoning capabilities in LLMs not by globally altering the entire output distribution, but by selectively adjusting the probabilities of a small subset of tokens that correspond to key reasoning behaviors. The majority of token probabilities — which encode core knowledge and reasoning content — remain largely unchanged.

Specifically, if RL primarily teaches models reasoning behaviors (*how* to reason) by modulating output probabilities rather than imparting fundamentally new knowledge or concepts (*what* to reason), then the adjustments learned through RL should inherently reflect reasoning skills already latent within base models. This reasoning-centric view suggests that these learned adjustments may not strongly depend on specific model scales or capacities. Consequently, this hypothesis presents a significant opportunity: applying RL to smaller, computationally efficient models and subsequently transferring the learned probabilistic adjustments to larger, more capable models. Such an approach could substantially mitigate the prohibitive computational and financial costs associated with directly performing RL on large-scale models.

We test our hypothesis via a preliminary study that compares the token-level decoding path shifts between base $\mathcal{M}_{\text{base}}$ and RL-trained models $\mathcal{M}_{\text{RL}}$ [1] of varying sizes. As illustrated in Figure 1(b), $\mathcal{M}_{\text{RL}}$ introduces only minimal shifts in the decoding path, with around 96.71% of tokens remaining unchanged in this specific case. Notably, these shifts are highly localized to a few reasoning-critical tokens, such as those triggering self-verification, branching out, or backtracking. This suggests that RL acts by amplifying latent reasoning behaviors rather than rewriting entire outputs.

Inspired by the above findings, we propose a simple and intuitive method, RAST, which leverages shifts in the output space to transfer learned "reasoning patterns" from a small RL-trained model (relative to its base) to larger base models across different scales. Extensive experimental results show that RAST substantially and consistently enhances the reasoning capabilities of base models, while requiring significantly lower GPU memory than direct RL training. Surprisingly, sometimes RAST yield even better performance than the RL-trained counterparts. Additionally, RAST also increases search space diversity compared to conventional RL training, exemplified by the superior pass@k performance. We further conduct detailed analyses on why RAST works and provide insights and practical guidelines for applying RAST, hoping to shed light on future works in this research line.

2 Methodology

Our goal is to enable a large base model $\mathcal{M}_{\text{base}}$ to emulate the reasoning behavior of a smaller reasoner $\mathcal{S}_{\text{RL}}$ tuned with RL, without requiring expensive RL training at scale. To this end, we begin with a preliminary study that motivates our core hypothesis and then introduce *reasoning activation via small-model transfer*, termed as RAST, a simple yet effective decoding-time method that activates reasoning capabilities across model scales.

[1] Unless otherwise specified, all mentions of RL-trained models in this paper refer to "RL from scratch" [76], directly training models using RL from the base model without SFT warmup.

https://doi.org/10.52202/085713-1663

2.1 Preliminary Study

We begin with a preliminary study to empirically support our *hypothesis* from Sec. 1 — RL activates reasoning capabilities in LLMs by adjusting the probabilities of *a small set of key tokens* that relate to particular reasoning behaviors. Concretely, we take the decoding path $T = [t_1, t_2, ..., t_n]$ from $\mathcal{M}_{\text{RL}}$, and feed it into $\mathcal{M}_{\text{base}}$ token by token to see if the next token prediction t'_{i+1} aligns with t_{i+1} (i.e., $\mathcal{M}_{\text{base}}$ generates the t_{n+1}-th token based on the first n tokens generated by $\mathcal{M}_{\text{RL}}$, regardless of the previous $\mathcal{M}_{\text{base}}$ generated tokens):

$$t'_{i+1} = \arg\max_t P_{\mathcal{M}_{\text{base}}}(t \mid t_1, \ldots, t_i) \stackrel{?}{=} t_{i+1} \tag{1}$$

This setup enables us to quantify how likely $\mathcal{M}_{\text{base}}$ is to recover the RL path. This study is conducted under greedy decoding to avoid potential randomness. To capture this alignment quantitatively, we define *Path Coverage Rate (PCR)* as the proportion of tokens in T for which the base model exactly matches the RL output:

$$\text{PCR}(T) = \frac{1}{n-1} \sum_{i=1}^{n-1} \mathbb{I}\left[t'_{i+1} = t_{i+1}\right] \tag{2}$$

A high PCR indicates that the $\mathcal{M}_{\text{base}}$ is already well-aligned with the RL decoding path, with only minor adjustments needed to activate desired reasoning behaviors. In our implementation, Qwen-2.5-32B-SimpleRL-Zoo serves as $\mathcal{M}_{\text{RL}}$ and Qwen2.5-32B is used as $\mathcal{M}_{\text{base}}$. We randomly sampled 50 trajectories from $\mathcal{M}_{\text{RL}}$ from MATH500 [18] and took the sample-level average as the final results. As shown in Figure 1(a), we observe that PCR remains remarkably high ($> 95\%$) across all model scales, indicating that RL-induced distributional shifts are notable only on a small set of tokens, with the majority of tokens also predictable by $\mathcal{M}_{\text{base}}$. Additionally, we found that the disparities mainly come from tokens that reflect certain reasoning behaviors. This indicates that by steering around these key tokens, $\mathcal{M}_{\text{base}}$ is able to recover the reasoning path generated by the RL-trained model.

2.2 RAST: Reasoning Activation in LLMs via Small-model Transfer

Building on the findings from our preliminary study, we hypothesize that minor, targeted adjustments to the $\mathcal{M}_{\text{base}}$'s output distribution can effectively enable it to perform on par with its RL-trained counterpart, without the need for expensive RL optimization directly conducted upon the large base model. In other words, we aim to transform $\mathcal{M}_{\text{base}}$ into a stronger reasoner at inference time by activating the latent reasoning capabilities already present within it in the token/output space.

To this end, we propose RAST, a decoding-time method that activates reasoning capabilities in large models by transferring logit-level adjustments from smaller RL-tuned models. Given the smaller model pair $\mathcal{S}_{\text{base}}$ and $\mathcal{S}_{\text{RL}}$, we propose leveraging their differences in logit distributions as reusable reasoning correction signals. Specifically, at the decoding time stamp t, with previous input as $x_{<t}$, we compute the logit scores of $\mathcal{M}_{\text{base}}$, $\mathcal{S}_{\text{base}}$, $\mathcal{S}_{\text{RL}}$, and define the final probability distribution over tokens for the enhanced model $\tilde{\mathcal{M}}$ as:

$$P_{\tilde{\mathcal{M}}}(X_t \mid x_{<t}) = \text{softmax}\left[\mathcal{M}_{\text{base}}(X_t \mid x_{<t}) + \lambda(\mathcal{S}_{\text{RL}}(X_t \mid x_{<t}) - \mathcal{S}_{\text{base}}(X_t \mid x_{<t}))\right] \tag{3}$$

where the adjustment terms represent the difference in token-level scoring between the RL-trained model and its base counterpart, denoted as ΔR. λ controls the strengths of ΔR. These logits encode reasoning-oriented shifts that can be transferred to the larger base model, allowing it to mimic improved inference behavior without retraining. Figure 2 outlines the overview of RAST, showing how ΔR from a small RL-tuned model is injected to adjust the output distribution of $\mathcal{M}_{\text{base}}$ at decoding time, selectively amplifying reasoning-relevant tokens (e.g., "instead") while preserving base predictions (e.g., "of") elsewhere. The additive formulation enables lightweight adaptation at inference time by altering the output distribution in a way that reflects reasoning preferences learned by the smaller model.

Our method is conceptually related to [36, 37, 31], which apply similar manipulation in logits spaces for generation steering. However, while these methods are often task-specific or used for controlling style/toxicity, our focus is on eliciting complex reasoning behavior, and we demonstrate that even small-scale reasoning signals can be reliably transferred across model sizes and domains.

3 Unlocking Reasoning Activation

https://doi.org/10.52202/085713-1663

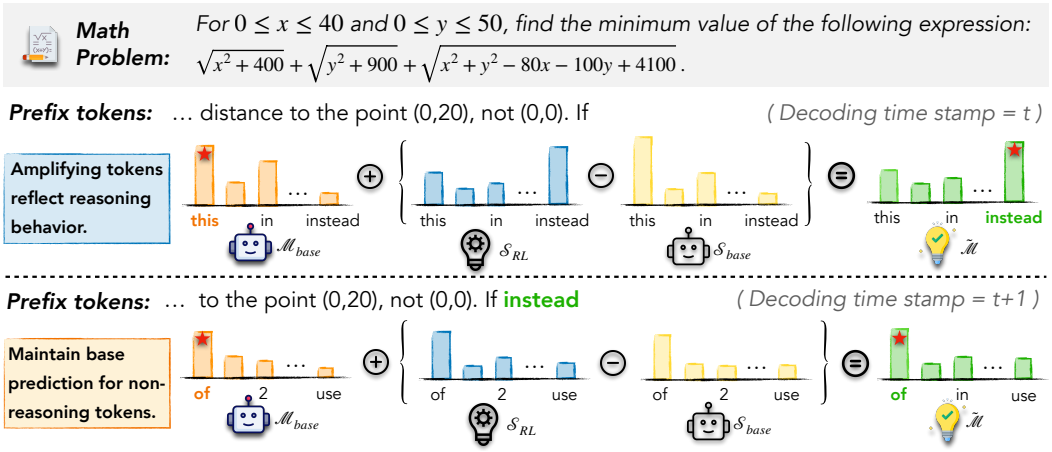


Figure 2: A concrete illustration of RAST: logit differences from a small RL-tuned model $\mathcal{S}_{\text{RL}}$ guide a large base model $\mathcal{M}_{\text{base}}$ at decoding time, amplifying reasoning-relevant predictions (e.g., "instead") while maintaining base outputs for non-reasoning tokens (e.g., "of").

3.1 Experimental Setup

Models and Tasks We systematically evaluate model performance across a comprehensive suite of mathematical reasoning tasks of varying difficulty, including standard benchmarks such as MATH500 [18], Minerva [30], OlympiadBench [17], GSM8K [6], as well as competition-level benchmarks AIME24 and AMC23. Our primary models are from the Qwen-2.5 family (1.5B, 7B, 14B, and 32B) [71] alongside their corresponding RL-trained variants using SimpleRL-Zoo [76] (e.g., SimpleRL-7B denotes the "RL from scratch" variant of Qwen-2.5-7B). We further demonstrate the generalizability of RAST across different model architectures and downstream tasks. To validate this, we conduct experiments using the Llama-3.1 series (8B, 70B) [13] and their zero RL-trained counterparts. Additionally, we assess model performance on coding tasks, utilizing zero RL-trained models from Code-R1 [40] trained from Qwen-2.5-1M [72] on coding benchmarks including HumanEval [2], MBPP+ [1], and LiveCodeBench [24]. For details of datasets and setup used in our experiments, please refer to Appendix A. We also present additional experiments on MMLU [18] and GPQA [54] and analysis results in Appendix C.

Evaluation Metrics Following previous work [19], and to ensure rigorous evaluation across models and tasks, we perform inference runs k up to 32 for each experimental setup and report the following metrics calculated over the collected trajectories:

- **Pass@k**: Pass@k evaluates whether at least one correct solution is found among k sampled outputs per problem. Formally,
$$\text{Pass@}k = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\sum_{j=1}^{k}\mathbb{I}\left(\hat{y}_{i,j}=y_i\right)\geq 1\right), \tag{4}$$
where N is the total number of problems, $\hat{y}_{i,j}$ is the prediction for the i-th problem in the j-th run, y_i is the corresponding ground truth, and $\mathbb{I}(\cdot)$ denotes the indicator function.[2]
- **Recovery Rate**: The recovery rate quantifies how much of the gap between the base model and a stronger RL-tuned model is recovered by the proposed method. Formally,
$$\text{Recovery Rate} = \frac{\text{Accuracy}_{\text{RAST}} - \text{Accuracy}_{\text{Base}}}{\text{Accuracy}_{\text{RL}} - \text{Accuracy}_{\text{Base}}}, \tag{5}$$
where "Accuracy" denotes the averaged pass@1 over 32 runs. Higher values of *recovery rate* indicate a more effective recovery of the performance gap.

Decoding Configurations To accelerate the inference speed, we implemented a revised vLLM [27] version to support RAST. For mathematical reasoning, decoding is performed using a temperature

[2]We use the same answer extraction and matching for performance evaluation as SimpleRL, following https://github.com/hkust-nlp/simpleRL-reason/tree/v1/examples/simplelr_math_eval.

Table 1: Experiment results of RAST on mathematical reasoning datasets with Qwen-2.5 model series. All numbers are computed across 32 runs with sampling, except that all base models use greedy decoding. *Avg.* indicates the averaged *pass@1* over 32 runs and *RR.* denotes the recovery rate.

Models	Math500		AIME24		AMC		Minerva		Olympiad		GSM8K	
	Avg.	*RR.*	*Avg.*	*RR.*	*Avg.*	*RR.*	*Avg.*	*RR.*	*Avg.*	*RR.*	*Avg.*	*RR.*
Qwen-2.5-32B	68.6	-	3.3	-	52.5	-	21.0	-	33.1	-	93.1	-
$+\Delta R_{1.5B}$ †	73.7	40.2%	12.5	37.9%	54.7	12.9%	25.1	84.1%	36.7	30.5%	93.3	7.7%
$+\Delta R_{7B}$	79.0	81.9%	16.4	53.9%	58.4	34.5%	32.0	87.3%	41.7	72.9%	94.4	50.0%
$+\Delta R_{14B}$	80.7	95.3%	18.3	61.7%	65.2	74.3%	34.2	104.8%	43.5	88.1%	95.3	84.6%
SimpleRL-32B	81.3	-	27.6	-	69.6	-	33.6	-	44.9	-	95.7	-
Qwen-2.5-14B	63.8	-	6.7	-	47.5	-	19.1	-	31.1	-	91.4	-
$+\Delta R_{1.5B}$ †	70.3	45.1%	10.9	54.5%	50.3	25.0%	22.9	29.0%	35.5	37.0%	92.4	31.3%
$+\Delta R_{7B}$	77.4	94.4%	16.2	123.4%	56.1	76.8%	28.8	74.0%	40.0	75.4%	93.3	59.4%
SimpleRL-14B	78.2	-	14.4	-	58.7	-	32.2	-	42.9	-	94.6	-
Qwen-2.5-7B	62.9	-	6.7	-	32.5	-	19.9	-	28.0	-	87.7	-
$+\Delta R_{1.5B}$ †	68.4	41.0%	9.5	31.5%	41.5	38.5%	23.0	51.7%	34.6	57.4%	89.8	50.0%
SimpleRL-7B	76.3	-	15.6	-	55.9	-	25.9	-	39.5	-	91.9	-

† indicates that the prompt used for RAST does not match the one used to train the small RL-tuned model (see Figure 7), due to the training inconsistency for available RL-tuned models (e.g., from SimpleRL-Zoo). As a result, the transferred $\Delta R_{1.5B}$ yields relatively smaller gains.

Table 2: Experiment results of RAST on mathematical reasoning datasets. ΔR is borrowed from Llama-3.1-8B-SimpleRL-Zoo [76]. Numbers are computed on 32 runs with sampling.

Models	Math500	AIME24	AMC	Minerva	Olympiad	GSM8K
Llama-3.1-70B	26.2	0.0	12.5	10.3	5.9	56.1
$+\Delta R_{8B}$	33.1	2.8	16.9	12.6	7.1	82.2

Table 3: Experiment results of RAST on code reasoning tasks with the Qwen-2.5-14B-1M model as $\mathcal{M}_{base}$ and ΔR from Code-R1-Zero [40] using greedy decoding.

Models	HumanEval+	MBPP+	LiveCodeBench	Average
Qwen-2.5-14B-1M	82.3	69.6	34.3	62.1
$+\Delta R_{7B}$	85.4	76.5	37.6	66.5

setting of 1.0 and nucleus sampling with a top-p of 0.95, allowing a maximum generation length of $16,384$ tokens, consistent with prior work [76]. We set λ in Equation 3 to 1.0 for all experiments. For code reasoning tasks, we follow previous evaluation settings [40, 24] and use greedy decoding. Specifically, we use EvalPlus [38, 39] for HumanEval+ and MBPP+. Our experiments are conducted over 8 NVIDIA A6000 GPUs on a single node, with GPU utilization and tensor parallelism parameters dynamically adjusted based on the model size. Typically, inference requires approximately 30 minutes per run for larger datasets like MATH500 and GSM8K, whereas smaller datasets such as AIME24 and AMC23 complete within 3–5 minutes per run. For specific parameters used for each benchmark, please refer to Appendix B.

3.2 Main Results

RAST enables consistent and scalable reasoning gains. Table 1 summarizes the performance of RAST on six mathematical reasoning benchmarks using the Qwen-2.5 model family at 1.5B, 7B, 14B, and 32B scales. Across all settings, RAST delivers substantial improvements over the base models in both averaged pass@1 and the corresponding recovery rate. Notably, with signals from smaller RL-trained models, RAST can even approach or beat the performance of RL-trained counterparts of the base model. For instance, applying ΔR_{14B} to the 32B base model achieves approximate or superior results on MATH500, Minerva, and GSM8K compared with the 32B RL-trained model. These findings validate the effectiveness of RAST in enhancing reasoning capabilities at inference time, without any retraining or RL on the target model.

https://doi.org/10.52202/085713-1663

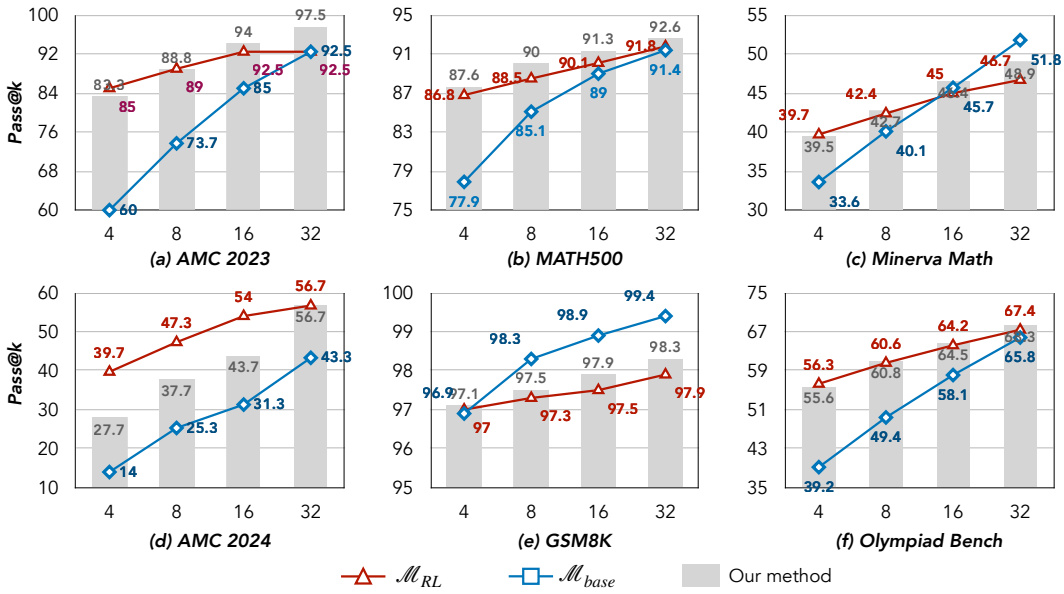


Figure 3: The illustration of pass@k for different values of k on 6 mathematical reasoning datasets, where $\mathcal{M}_{\text{base}}$ is Qwen-2.5-32B, RAST uses ΔR_{14B}, and $\mathcal{M}_{\text{RL}}$ is the RL-trained version of $\mathcal{M}_{\text{base}}$.

ΔR from stronger experts yield greater gains. The effectiveness of RAST also depends on the strength of the $\mathcal{S}_{\text{RL}}$ and $\mathcal{S}_{\text{base}}$ that generates the delta logit ΔR. For each base model, using larger delta sources (e.g., base model of 32B with ΔR_{7B} or ΔR_{14B} compared with $\Delta R_{1.5B}$) leads to greater improvement. Taking the 32B base on MATH500 as an example, accuracy increases progressively from 73.7 (with $\Delta R_{1.5B}$) to 80.7 (with ΔR_{14B}), while the ceiling model, or the upper bound reaches 81.3. Similarly, ΔR_{7B} also works better than $\Delta R_{1.5B}$ for the 14B base model across all datasets. This trend suggests that logit deltas encode richer reasoning signals as the $\mathcal{S}_{RL}$ model scale increases, making RAST a flexible tool for knowledge transfer across various model scales.

Trade-off between $\mathcal{M}_{\text{base}}$ and ΔR. The effectiveness of RAST also depends on the capacity of the base model $\mathcal{M}_{\text{base}}$ and its alignment with ΔR. In general, stronger base models exhibit higher recovery rates, indicating greater receptiveness to transferred reasoning signals. For example, when applying RAST to the 32B base, the recovery rate is often higher than when using 14B or 7B. On GSM8K, RAST bridges nearly the entire gap between the base model (93.1) and the RL expert (95.7), achieving 95.3 with the delta logit from a 14B model. In contrast, for the 7B base, the gain is smaller (87.7 to 91.9), even though the same ΔR_{7B} is applied. This suggests that higher-capacity base models are more receptive to the transferred reasoning signal. However, increasing base model capacity alone does not guarantee better outcomes. When applying ΔR_{7B} to both 14B and 32B bases, the 14B base yields a higher recovery rate, suggesting that a large capability gap between the base and ΔR may hinder effective transfer. These observations highlight a trade-off: while stronger base models benefit more from compatible deltas, excessively mismatched pairs may reduce the efficacy of reasoning activation.

3.3 RAST Boosts Reasoning Diversity

Figure 3 illustrates the pass@k accuracy trends across six mathematical reasoning benchmarks, using the Qwen-2.5-32B base model augmented with ΔR_{14B}. For each problem, we randomly sample k outputs from a 32-sample pool, repeat this process 10 times, and report the average accuracy. This setup allows us to assess how well RAST supports diverse solution trajectories under varying sampling sizes. Based on the results, we have the following key observations:

Increasing k consistently improves accuracy. Across all benchmarks, pass@k increases monotonically with larger k, confirming the benefit of sampling multiple decoding paths. This trend reflects that RAST promotes solution diversity—a larger k enables broader exploration of plausible answers, increasing the likelihood of capturing correct responses even when individual generations are imperfect. We also noticed that on most benchmarks except GSM8K, pass@k for $\mathcal{M}_{\text{RL}}$ is large than $\mathcal{M}_{\text{base}}$, which might result from GSM8K's simplicity.

https://doi.org/10.52202/085713-1663

Pass@k surpasses the ceiling performance. Remarkably, in all benchmarks, RAST achieves pass@k accuracy that equals or even exceeds the performance of $\mathcal{M}_{\text{RL}}$. This stands in contrast to prior findings [57, 74], which reported limited or deteriorated performance in pass@k under RL training. We posit that this effect may stem from the implicit ensembling of knowledge across models, which enhances diversity in the search space. Additionally, RAST can sometimes beat the pass@k of $\mathcal{M}_{\text{base}}$ on benchmarks like AMC, MATH500, and Olympiad Bench. This surprising behavior suggests that RAST not only saves the costly training efforts of RL, but also introduces a distinct form of diversity or guidance in the sampling space that helps recover correct answers, a notable drawback of RL-trained models.

3.4 RAST Generalizes Well to Other Models and Tasks

We conduct experiments of RAST on another model family, Llama-3.1, and on additional code reasoning tasks. The results are shown in Tables 2 and 3. Applying RAST to Llama-3.1-70B using ΔR_{8B} from a smaller RL-tuned model [76] yields consistent gains across all six mathematical reasoning datasets. As shown in Table 2, we observe a $+2.8$ absolute improvement on AIME24, $+4.4$ on AMC, and $+2.3$ on Olympiad. We also found that the improvement in MATH500 and GSM8K was particularly high. This might be due to the training recipe of S_{RL}, where they are trained with problems from MATH500 and GSM8K. Nonetheless, the experiments demonstrate that RAST is effective for different model families apart from Qwen, reaffirming the hypothesis that reasoning-relevant distributional shifts are transferrable and model-agnostic. In Table 3, we evaluate RAST on Qwen-2.5-14B-1M [72] using ΔR from its 7B counterpart [40] on three code reasoning benchmarks. RAST achieves consistent improvements on all datasets, leading to an overall +4.4 absolute improvement in average performance. This indicates that the method is not limited to mathematical reasoning, but code-related reasoning tasks. These findings confirm the broad applicability of RAST across model architectures, parameter scales, and reasoning domains.

4 Understanding Reasoning Activation

4.1 Similarity of ΔR as Signal for Transferability

As shown in Table 1, model performance varies widely across settings, motivating the need to understand what governs effective transferability. The delta logits ΔR (defined in Equation 3) capture the modification induced by the reasoning-enhanced model relative to its base. To quantify the alignment between these delta logits across different model pairs, we adopt *cosine similarity* as the measure. This choice is motivated by prior work [61, 15], which demonstrates that cosine similarity effectively captures directional alignment in high-dimensional spaces, independent of magnitude. Following Section 2, we randomly sample 50 examples from MATH500 and extract the decoding trajectory T (with tokens t) generated by SimpleRL-32B. Each trajectory is then fed through $\mathcal{M}_{\text{base}}$ and $\mathcal{M}_{\text{RL}}$ token by token to compute ΔR with prefix $T_{<t}$. We then log the average cosine similarity across the trajectory as:

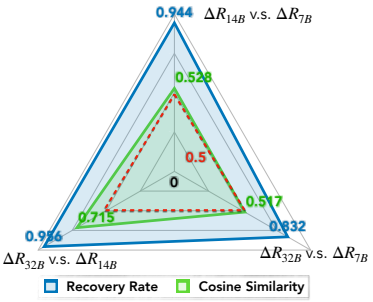


Figure 4: Cosine similarity vs. recovery rate across delta logit pairs (ΔR) from varying model scales. E.g., "ΔR_{14B} v.s. ΔR_{7B}" denotes AvgCosineSim($\Delta R_{14B}, \Delta R_{7B}$).

$$\text{AvgCosineSim}(\Delta R_1, \Delta R_2) = \frac{1}{T}\sum_{t=1}^{T} \frac{\Delta R_1^{(t)} \cdot \Delta R_2^{(t)}}{\|\Delta R_1^{(t)}\| \, \|\Delta R_2^{(t)}\|} \tag{6}$$

The results are shown in Figure 4. We found that the recovery rate exhibits a positive correlation with cosine similarity—as ΔR between models becomes more aligned, transferability improves.

4.2 Empirical Token-Level Signals of Reasoning Activation

To further examine how RAST elicits reasoning behaviors during generation, we perform a token-level analysis focusing on linguistic traces that reflect three hallmark reasoning behaviors: (i) *branching out*, (ii) *backtracking*, and (iii) *self-verification* as previously mentioned in Section 1. For each behavior,

https://doi.org/10.52202/085713-1663

we manually curated a set of representative tokens based on prior qualitative observations and related work [11, 52, 63]. These tokens serve as behavioral signatures that may surface when the model engages in complex reasoning. The complete set of curated tokens is displayed in Figure 9. We also present a case study of token-level reasoning activation for a more intuitive view in Appendix C.5.

We compute the frequency of these tokens in all 32 model output trajectories across six mathematical reasoning benchmarks, comparing the base model $\mathcal{M}_{base}$, RL-trained ceiling model $\mathcal{M}_{RL}$, and RAST with ΔR_{14b} applying to 32B base models. Figure 5 presents the normalized occurrence rates of each token category across different models. Two key trends emerge:

Dataset-specific reasoning emphasis. Different benchmarks accentuate different types of reasoning behaviors. For instance, AIME 2024 and Olympiad Bench exhibit a pronounced increase in *branching out* and *backtracking* tokens, indicating that these tasks may demand exploring alternative solution paths and revisiting previous steps. In contrast, datasets like AMC 2023 and MATH500 emphasize *self-verification*, with higher frequencies of verification-related tokens such as "check" or "confirm". This variation suggests that reasoning demands are not uniform across benchmarks, and token-level analysis can surface behavior-specific task signals.

RAST steers base models closer to RL-trained behaviors. Across all datasets and reasoning categories, we observe that RAST consistently increases the frequency of reasoning-related tokens relative to $\mathcal{M}_{base}$ and more closely matches the RL-trained model $\mathcal{M}_{RL}$. This alignment is particularly clear in Minerva, Olympiad Bench, and GSM8K, where RAST nearly mirrors the behavior of $\mathcal{M}_{RL}$ in the *self-verification* dimension. These results support our central claim that the delta logit signal ΔR effectively induces reasoning traits without the need for full RL training on large-scale models.

Together, these findings offer empirical evidence that RAST not only activates latent reasoning capabilities within base models but also tailors such activation in a task-sensitive manner, approximating the behavioral signature of much costlier RL-trained experts.

4.3 Efficiency Analysis

GPU Memory Overhead. We first present the efficiency analysis in terms of estimated GPU memory requirements, highlighting the computational advantage of RAST over conventional RL training. We summarize the results in Table 4. We estimate the memory overhead in terms of GPU memory used for RAST (including training $\mathcal{S}_{\text{RL}}$ and inference cost), and $\mathcal{M}_{\text{RL}}$. The results are estimated considering tensor parallel and CPU offloading (since these are common tricks during training) based on the following dimensions: (i) model memory footprint (FP16), (ii) optimizer states, and (iii) activations & buffers. Details of the computation for estimation could be found in Appendix D. Despite using significantly fewer resources, RAST achieves high recovery rates across all settings (e.g., reaching over 84% in the most demanding 32B + ΔR_{14B} configuration). This demonstrates that RAST retains most of the performance benefits of full-scale RL training while reducing the computational burden by up to 50% in terms of GPU memory and hardware requirements.

Table 4: Comparison of memory overhead and inference speed between our approach RAST and the conventional RL training pipeline (i.e., GRPO). We also report the averaged performance recovery rate (RR.) for all mathematical reasoning benchmarks.

Settings	**32B**		**32B**		**14B**	
	$\mathcal{M}_{\text{base}}$	$\mathcal{M}_{\text{RL}}$	$\mathcal{M}_{\text{base}}$	$\mathcal{M}_{\text{RL}}$	$\mathcal{M}_{\text{base}}$	$\mathcal{M}_{\text{RL}}$
	$+\Delta R_{14B}$	-	$+\Delta R_{7B}$	-	$+\Delta R_{7B}$	-
Memory Overhead						
GPU memory	∼160G	∼350G	∼100G	∼350G	∼100G	∼160G
# GPU cards	4×80G	8×80G	2×80G	8×80G	2×80G	4×80G
Inference Speed						
Input Tokens	61.1	73.0	62.4	73.0	65.0	75.1
Output Tokens	200.3	243.1	202.7	243.1	216.1	249.5
Averaged RR.	84.8%	100%	63.4%	100%	83.9%	100%

Inference Speed. Table 4 further offers the actual inference speed in terms of the number of input and output tokens processed per second for RAST's practicality. We observe that RAST incurs around 13-16% latency at inference time compared to directly using $\mathcal{M}_{RL}$ for decoding at different scales. This is expected and acceptable due to the need to compute multiple model logits at each decoding step. We believe that this latency could be further optimized by efficient decoding methods such as speculative decoding [29, 69, 49].

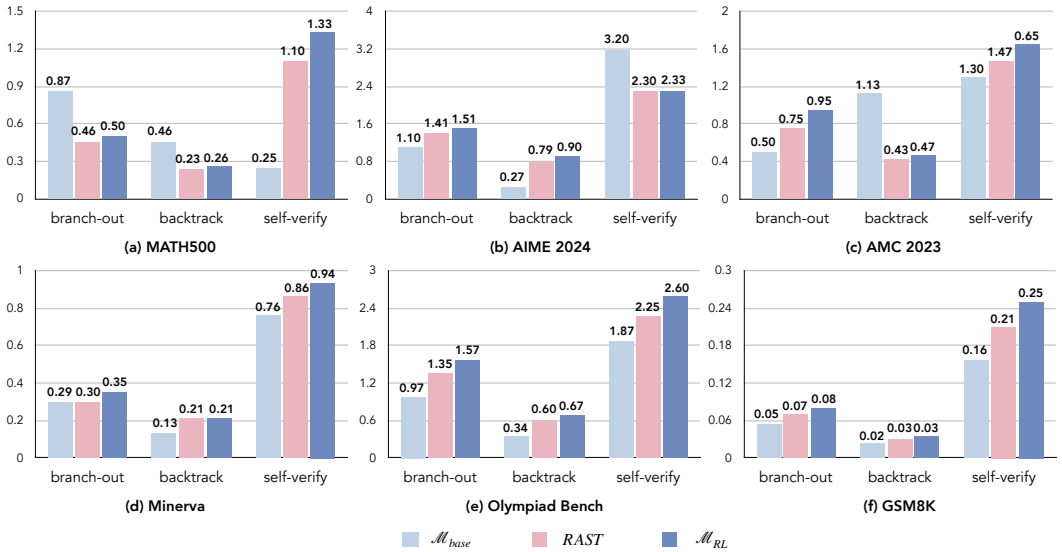


Figure 5: Normalized frequencies of reasoning-related tokens across three models ($\mathcal{M}_{\text{base}}$, RAST, and $\mathcal{M}_{\text{RL}}$) over six benchmarks. Each subfigure reflects a different reasoning behavior category.

4.4 Robustness regarding τ and λ

To evaluate the sensitivity of our method to decoding-time hyperparameters, we perform a grid search over sampling temperature τ and λ used in Equation 3. The results are summarized with performance trends visualized in Figure 6. We plot accuracy across varying τ values (with $\lambda = 0.5$ fixed) and varying λ values (with $\tau = 1.0$ fixed), including standard deviation error bars to reflect stability across multiple runs. Note that we use these fixed values for investigation since they represent the peak performance. In both settings, the accuracy remains consistently high within a reasonable range. Specifically, with $\tau \in [0.5, 1.0]$ and $\lambda \in [0.3, 1.5]$, the performance would be reasonably good within a certain range. These trends demonstrate that our method is robust to moderate fluctuations in decoding-time hyperparameters and does not rely on precise tuning for strong performance.

5 Related Work

5.1 Reinforcement Learning for LLM Complex Reasoning

Reinforcement learning (RL) has emerged as an important process of LLM post-training for human preference alignment. Early successes such as reinforcement learning with human feedback (RLHF) [48] demonstrated the effectiveness of reward modeling from preference data [75, 7]. Building upon this, follow-up works have explored RL as a means to directly optimize for reasoning quality [23] with outcome supervision [28] or process supervision [33, 64] as verifiable rewards, specifically in the domain of math [57] and coding [66, 40].

More recently, the success of DeepSeek-R1 [14] introduced a notable shift in training methodology with the "zero-RL" paradigm, where RL is applied directly to the base LLM, entirely bypassing intermediate supervised fine-tuning. Following its release, the open-source community has made significant strides in replicating [76, 50, 81], extending [73], and interpreting [78, 74, 42] the R1 algorithm and its behavioral consequences. Our work builds directly upon these open-source Zero-RL models and takes a step further by exploring whether the reasoning behaviors elicited through RL in small models can be transferred to larger base models without additional RL training. Specifically, we focus on leveraging the output distribution (logits) of small RL-trained reasoning models to activate similar behaviors in larger models across scales.

5.2 Decoding-time Strategy in Language Models

Decoding-time methods [59] have been explored largely in text generation, typically by manipulating the logit distribution from base language models. Earlier research efforts focused on sampling-based

https://doi.org/10.52202/085713-1663

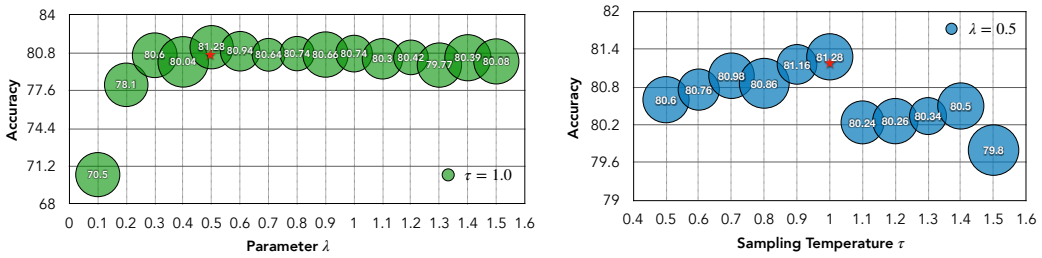


Figure 6: Experiment results with varying λ (left) and τ (right) on MATH500 dataset, where ★ denotes the peak performance. The position represents the accuracy, and the size of the circle denotes the standard deviation over 32 runs.

strategies aimed at improving generation quality through techniques like hierarchical decoding [10], nucleus sampling [20], and locally typical sampling [45]. More recent approaches incorporate the notion of *contrastiveness*. Specifically, Contrastive Decoding [31] exploits the disagreement between a stronger (expert) and a weaker (amateur) model to down-weight completions favored by the weaker model. DExperts [37] steers generation by contrasting models fine-tuned on desirable and undesirable attributes. Proxy-tuning [36] and emulated tuning [46] further leverage the idea for training-free alignment, while DoLA [5] contrasts different layers in next-token-prediction for factuality. The philosophy of contrastiveness has since been extended to a variety of downstream tasks, including machine translation [65], retrieval-augmented generation (RAG) [53], and conflict resolution in knowledge-intensive tasks [60]. While these works primarily focus on stylistic control or factual consistency in open-ended text generation, our approach targets reasoning-specific improvements.

When it comes to reasoning, [47] explores contrastive decoding for mathematical reasoning, while [35] identifies critical tokens in the reasoning trajectory using contrastive estimation. Our work builds on this growing line of decoding-time reasoning enhancement. Specifically, we leverage the divergence between Zero-RL-trained experts and their base counterparts to guide decoding, aiming to surface and amplify reasoning behaviors. In contrast to prior approaches that operate on instruction-tuned or supervised models, our method explicitly targets RL-induced reasoning signals and explores their transferability across model scales.

6 Conclusion and Discussion

We introduce RAST, a decoding-time framework that enables scalable reasoning enhancement in LLMs by transferring logit-level guidance from smaller RL-tuned models. Comprehensive experiments show that RAST consistently boosts the performance, often approaching or surpassing the performance of much more expensive ceiling models. Further analysis confirms the robustness of our method across decoding hyperparameters and highlights the alignment between delta signals and reasoning activation. Our findings suggest a practical and efficient pathway for eliciting complex reasoning in LLMs, opening new avenues for decoding-time reasoning enhancement and model behavior study for RL. We also present some insights toward future directions in Appendix E.

Acknowledgments and Disclosure of Funding

We sincerely thank the anonymous reviewers and the area chair for their constructive feedback and valuable suggestions, which greatly helped improve this work. Research was supported in part by National Science Foundation IIS-19-56151, NSF IIS 25-37827, the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897, and the Institute for Geospatial Understanding through an Integrative Discovery Environment (I-GUIDE) by NSF under BRIES Program No. HR0011-24-3-0325. The research has used the Delta/DeltaAI advanced computing and data resource, supported in part by the University of Illinois Urbana-Champaign and through allocation #250851 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants OAC 2320345, #2138259, #2138286, #2138307, #2137603, and #2138296. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA or the U.S. Government.

References

[1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *ArXiv preprint*, abs/2108.07732, 2021.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021.

[3] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *ArXiv preprint*, abs/2412.21187, 2024.

[4] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Sergey Levine, and Yi Ma. SFT memorizes, RL generalizes: A comparative study of foundation model post-training. In *The Second Conference on Parsimony and Learning (Recent Spotlight Track)*, 2025.

[5] Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

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

[7] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback, 2024.

[8] Xingyu Dang, Christina Baek, J Zico Kolter, and Aditi Raghunathan. Assessing diversity collapse in reasoning. In *Scaling Self-Improving Foundation Models without Human Supervision*, 2025.

[9] Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, MCS '00, page 1–15, Berlin, Heidelberg, 2000. Springer-Verlag.

[10] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, 2018. Association for Computational Linguistics.

[11] Kanishk Gandhi, Ayush Chakravarthy, Anikait Singh, Nathan Lile, and Noah D Goodman. Cognitive behaviors that enable self-improving reasoners, or, four habits of highly effective stars. *ArXiv preprint*, abs/2503.01307, 2025.

[12] TNG Technology Consulting GmbH. Deepseek-r1t-chimera, 2025.

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

 https://doi.org/10.52202/085713-1663

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

[15] Gyeongdo Ham, Seonghak Kim, Suin Lee, Jae-Hyeok Lee, and Daeshik Kim. Cosine similarity knowledge distillation for individual class information transfer. *ArXiv preprint*, abs/2311.14307, 2023.

[16] Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning. *ArXiv preprint*, abs/2403.04642, 2024.

[17] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[18] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[19] Andreas Hochlehnert, Hardik Bhatnagar, Vishaal Udandarao, Samuel Albanie, Ameya Prabhu, and Matthias Bethge. A sober look at progress in language model reasoning: Pitfalls and paths to reproducibility. *ArXiv preprint*, abs/2504.07086, 2025.

[20] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[22] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic loRA composition. In *First Conference on Language Modeling*, 2024.

[23] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *ArXiv preprint*, abs/2412.16720, 2024.

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

[25] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[26] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[27] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[28] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. T\" ulu 3: Pushing frontiers in open language model post-training. *ArXiv preprint*, abs/2411.15124, 2024.

[29] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR, 2023.

[30] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[31] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12286–12312, Toronto, Canada, 2023. Association for Computational Linguistics.

[32] Eric Liang, Zhanghao Wu, Michael Luo, Sven Mika, Joseph E. Gonzalez, and Ion Stoica. Rllib flow: Distributed reinforcement learning is a dataflow problem. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 5506–5517, 2021.

[33] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[34] Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Raghavi Chandu, Chandra Bhagavatula, and Yejin Choi. The unlocking spell on base llms: Rethinking alignment via in-context learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[35] Zicheng Lin, Tian Liang, Jiahao Xu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. Critical tokens matter: Token-level contrastive estimation enhence llm's reasoning capability. *ArXiv preprint*, abs/2411.19943, 2024.

[36] Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy. In *First Conference on Language Modeling*, 2024.

[37] Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-time controlled text generation with experts and anti-experts. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online, 2021. Association for Computational Linguistics.

[38] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

 https://doi.org/10.52202/085713-1663

[39] Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. Evaluating language models for efficient code generation. In *First Conference on Language Modeling*, 2024.

[40] Jiawei Liu and Lingming Zhang. Code-r1: Reproducing r1 for code with reliable rewards. 2025.

[41] Zichen Liu, Changyu Chen, Wenjun Li, Tianyu Pang, Chao Du, and Min Lin. There may not be aha moment in r1-zero-like training — a pilot study. https://oatllm.notion.site/oat-zero, 2025. Notion Blog.

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

[43] Xinyin Ma, Guangnian Wan, Runpeng Yu, Gongfan Fang, and Xinchao Wang. Cot-valve: Length-compressible chain-of-thought tuning. *ArXiv preprint*, abs/2502.09601, 2025.

[44] Sara Vera Marjanović, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, et al. Deepseek-r1 thoughtology: Let's< think> about llm reasoning. *ArXiv preprint*, abs/2504.07128, 2025.

[45] Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121, 2023.

[46] Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D. Manning. An emulator for fine-tuning large language models using small language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[47] Sean O'Brien and Mike Lewis. Contrastive decoding improves reasoning in large language models. *ArXiv preprint*, abs/2309.09117, 2023.

[48] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[49] Siru Ouyang, Shuohang Wang, Minhao Jiang, Ming Zhong, Donghan Yu, Jiawei Han, and Yelong Shen. Temperature-centric investigation of speculative decoding with knowledge distillation. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13125–13137, Miami, Florida, USA, 2024. Association for Computational Linguistics.

[50] Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. Tinyzero. https://github.com/Jiayi-Pan/TinyZero, 2025. Accessed: 2025-01-24.

[51] Richard Yuanzhe Pang, Weizhe Yuan, He He, Kyunghyun Cho, Sainbayar Sukhbaatar, and Jason Weston. Iterative reasoning preference optimization. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[52] Tian Qin, David Alvarez-Melis, Samy Jelassi, and Eran Malach. To backtrack or not to backtrack: When sequential search limits model reasoning. *ArXiv preprint*, abs/2504.07052, 2025.

[53] Zexuan Qiu, Zijing Ou, Bin Wu, Jingjing Li, Aiwei Liu, and Irwin King. Entropy-based decoding for retrieval-augmented large language models. 2025.

[54] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[55] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv preprint*, abs/1707.06347, 2017.

[56] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[57] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[58] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient RLHF framework. In *Proceedings of the Twentieth European Conference on Computer Systems, EuroSys 2025, Rotterdam, The Netherlands, 30 March 2025 - 3 April 2025*, pages 1279–1297. ACM, 2025.

[59] Chufan Shi, Haoran Yang, Deng Cai, Zhisong Zhang, Yifan Wang, Yujiu Yang, and Wai Lam. A thorough examination of decoding methods in the era of LLMs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8601–8629, Miami, Florida, USA, 2024. Association for Computational Linguistics.

[60] Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 783–791, Mexico City, Mexico, 2024. Association for Computational Linguistics.

[61] Sungho Shin, Joosoon Lee, Junseok Lee, Yeonguk Yu, and Kyoobin Lee. Teaching where to look: Attention similarity knowledge distillation for low resolution face recognition. In *Computer Vision – ECCV 2022*, pages 631–647, Cham, 2022. Springer Nature Switzerland.

[62] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[63] Gokul Swamy, Sanjiban Choudhury, Wen Sun, Zhiwei Steven Wu, and J Andrew Bagnell. All roads lead to likelihood: The value of reinforcement learning in fine-tuning. *ArXiv preprint*, abs/2503.01067, 2025.

[64] Luong Trung, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. ReFT: Reasoning with reinforced fine-tuning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7601–7614, Bangkok, Thailand, 2024. Association for Computational Linguistics.

[65] Jonas Waldendorf, Barry Haddow, and Alexandra Birch. Contrastive decoding reduces hallucinations in large multilingual machine translation models. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2539, St. Julian's, Malta, 2024. Association for Computational Linguistics.

[66] Junqiao Wang, Zeng Zhang, Yangfan He, Yuyang Song, Tianyu Shi, Yuchen Li, Hengyuan Xu, Kunyu Wu, Guangwu Qian, Qiuwu Chen, et al. Enhancing code llms with reinforcement learning in code generation. *ArXiv preprint*, abs/2412.20367, 2024.

[67] Yuyang Wu, Yifei Wang, Tianqi Du, Stefanie Jegelka, and Yisen Wang. When more is less: Understanding chain-of-thought length in llms. *ArXiv preprint*, abs/2502.07266, 2025.

 https://doi.org/10.52202/085713-1663

[68] Heming Xia, Yongqi Li, Chak Tou Leong, Wenjie Wang, and Wenjie Li. Tokenskip: Controllable chain-of-thought compression in llms, 2025.

[69] Zilin Xiao, Hongming Zhang, Tao Ge, Siru Ouyang, Vicente Ordonez, and Dong Yu. Parallel-spec: Parallel drafter for efficient speculative decoding. 2025.

[70] Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning. *ArXiv preprint*, abs/2502.14768, 2025.

[71] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *ArXiv preprint*, abs/2412.15115, 2024.

[72] An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, et al. Qwen2. 5-1m technical report. *ArXiv preprint*, abs/2501.15383, 2025.

[73] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *ArXiv preprint*, abs/2503.14476, 2025.

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

[75] Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. Acecoder: Acing coder rl via automated test-case synthesis. *ArXiv preprint*, abs/2502.01718, 2025.

[76] Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild. *ArXiv preprint*, abs/2503.18892, 2025.

[77] Jinghan Zhang, Xiting Wang, Fengran Mo, Yeyang Zhou, Wanfu Gao, and Kunpeng Liu. Entropy-based exploration conduction for multi-step reasoning. *ArXiv preprint*, abs/2503.15848, 2025.

[78] Rosie Zhao, Alexandru Meterez, Sham Kakade, Cengiz Pehlevan, Samy Jelassi, and Eran Malach. Echo chamber: Rl post-training amplifies behaviors learned in pretraining. *ArXiv preprint*, abs/2504.07912, 2025.

[79] Yaowei Zheng, Junting Lu, Shenzhi Wang, Zhangchi Feng, Dongdong Kuang, and Yuwen Xiong. Easyr1: An efficient, scalable, multi-modality rl training framework. https://github.com/hiyouga/EasyR1, 2025.

[80] Ming Zhong, Chenxin An, Weizhu Chen, Jiawei Han, and Pengcheng He. Seeking neural nuggets: Knowledge transfer in large language models from a parametric perspective. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[81] Hengguang Zhou, Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. R1-zero's" aha moment" in visual reasoning on a 2b non-sft model. *ArXiv preprint*, abs/2503.05132, 2025.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims have been validated with extensive empirical results.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations in Section 6 and Appendix F.

https://doi.org/10.52202/085713-1663

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We released our code, and provided experimental details to reproduce our results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All the datasets used in our paper are open-access, and we also provide an anonymous GitHub repository for our code in the abstract.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

 https://doi.org/10.52202/085713-1663

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We presented experiment settings and details in Section 3.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All of our experimental results are computed on 32 runs, and we also provide the standard deviation for evaluation in Section 4.3 for statistical significance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We mentioned our computation environment to reproduce the experiments in Section 3.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research was conducted in accordance with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work is a methodological paper to activate reasoning capabilities in a large model with small-model guidance, and does not have direct societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

https://doi.org/10.52202/085713-1663

Justification: Our work does not involve releasing data or models that have a high risk for misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We properly credited the creators of the assets used in the paper, and provide URLs to all assets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We have released our code, used datasets, and inferenced trajectories, and provided detailed documentation in the anonymous GitHub repository mentioned in the Abstract.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve crowd-sourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Our work only uses LLMs for writing, editing, and formatting purposes that do not impact the core methodology, scientific rigor, and originality of the research.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

https://doi.org/10.52202/085713-1663

A Implementation Details

A.1 Datasets

We provide the details for all the datasets used in our work as follows. All datasets or benchmarks used in this paper are publicly available online. For mathematical reasoning tasks, we include 6 widely used datasets, detailed below:

MATH500 The original MATH collection contains 12,500 problems in total, with 8,000 training and 4,500 test problems, meticulously curated to cover a wide range of topics and difficulty levels. Each problem in MATH has a full step-by-step solution that can be used to teach models to generate answer derivations and explanations. MATH500 (could be found at https://huggingface.co/datasets/HuggingFaceH4/MATH-500) is a non-standard train/test split of the original MATH dataset [18], following [33] to avoid the risk of over-fitting and for more efficient testing configurations. These 500 test problems are selected uniformly at random, and are representative of the test set as a whole.

GSM8K GSM8K (Grade School Math 8K) [6] is a dataset of 8,500 high-quality linguistically diverse grade school math word problems. The dataset was created to support the task of question answering on basic mathematical problems that require multi-step reasoning. The test set of GSM8K (could be found at https://huggingface.co/datasets/openai/gsm8k) includes 1,319 problems in total.

Olympiad Bench Olympiad Bench [17] is originally an Olympiad-level bilingual multimodal scientific benchmark, which contains 8,952 math and physics questions from international Olympiads, Chinese Olympiads, Chinese college entrance examinations, and mock exams. To support our testing needs, we select a subset from the Olympiad Bench that is categorized as "open-ended", "text-only" and "competition-level". Together, there are 675 test problems (could be found at https://huggingface.co/datasets/Hothan/OlympiadBench/viewer/OE_TO_maths_en_COMP) for this subset used in our paper.

AIME The collection of AIME actually contains problems from the American Invitational Mathematics Examination (AIME). AIME is a prestigious high school mathematics competition known for its challenging mathematical problems. In our work, we follow previous works and adopt all the 30 problems (could be found at https://huggingface.co/datasets/HuggingFaceH4/aime_2024) in AIME 2024 for testing purposes.

AMC Similar to AIME, AMC is another very challenging dataset that contain problems in competitions, specifically, the American Mathematics Competitions (AMC). The collection of AMC actually contains 40 problems (could be found at https://huggingface.co/datasets/math-ai/amc23) in total in the year of 2023 for our testing set.

Minerva The testing set of Minerva math is curated in [30], which consists of STEM problems at the undergraduate level. In total, there are 272 problems (could be found at https://huggingface.co/datasets/math-ai/minervamath), 191 of which have numeric solutions and 81 have symbolic solutions.

For code reasoning tasks, we incorporate three datasets as following:

HumanEval+ HumanEval+ is an adapted version from the original HumanEval by [38]. HumanEval+ extends beyond HumanEval with additional high-quality and automatically generated test inputs to 80×, powered by both LLM- and mutation-based strategies. It contains 164 samples for testing (could be found at https://huggingface.co/datasets/evalplus/humanevalplus).

MBPP+ HumanEval+ is also an adapted version from the original MBPP by [38]. The construction process is quite similar to HumanEval+, and it contains 378 samples for the testset (could be found at https://huggingface.co/datasets/evalplus/mbppplus).

Prompt (a)

<lim_startl>**system**
You are a helpful assistant.<lim_endl>
<lim_startl>**user**
{input}
Please reason step by step, and put your final answer within \\boxed{{}}.<lim_endl>
<lim_startl>assistant

Prompt (b)

Question:\n*{input}*\n**Answer:**\nLet's think step by step.

Figure 7: Prompt templates used for mathematical reasoning tasks.

Table 5: Detailed decoding configurations (hyperparameters) used in our experiments.

Settings	gpu_utilization	tensor_parallel_size	temperature	λ	top_p	max_seq_len
Qwen-2.5-32B	0.6	4	0.0	1.0	1.0	16,384
+$\Delta R_{1.5B}$	0.6	4	1.0	1.0	0.95	
+ΔR_{7B}	0.75	4				
+ΔR_{14B}	0.75	4				
32B-RLZero	0.6	4				
Qwen-2.5-14B	0.6	2	0.0	1.0	1.0	16,384
+$\Delta R_{1.5B}$	0.6	2	1.0	1.0	0.95	
+ΔR_{7B}	0.75	2				
14B-RLZero	0.6	2				
Qwen-2.5-7B	0.6	1	0.0	1.0	1.0	16,384
+$\Delta R_{1.5B}$	0.6	1	1.0	1.0	0.95	
7B-RLZero	0.6	1				

LiveCodeBench LiveCodeBench [24] is a recently proposed benchmark that aims at a comprehensive and contamination-free evaluation of LLMs for code, which continuously collects new problems over time from contests across three competition platforms, namely LeetCode, AtCoder, and CodeForces. In our experiments, there are 880 test samples in total.

A.2 Prompt Templates for Inference

Inference prompts used in this work generally following the common practice in the open-source community. Specifically for mathematical reasoning tasks, there are two kinds of prompts as shown in Figure 7. During our experiments, we select experiments with respect to ΔR. For $\Delta R_{1.5B}$, we choose to use *Prompt (b)* following the training and inference setting in [76] since the model scale is too small to follow the complex chat template and output format instruction of "\boxed". For both prompt templates, the "{input}" will be substituted with the corresponding input problem for each data sample.

B Decoding Configurations

Table 5 presents the detailed decoding configurations for our experiments, specifically, the hyperparameters used.

https://doi.org/10.52202/085713-1663

Table 6: PCR results on various settings.

Backbone Models	AIME	MATH500	GPQA	MMLU
Qwen-7B	96.03	96.67	93.73	94.54
Qwen-14B	95.60	96.28	93.67	93.91
Qwen-32B	95.22	96.58	92.93	93.32
Llama-8B	91.15	93.16	93.23	90.44

C More Results

C.1 Additional PCR Results

We provide a comprehensive view of PCR results on different datasets, model families, and scales used across this paper in Table 6.

We can see that our conclusion still holds, with over 90 PCRs across all the different model families and different benchmark data points. The results prove the generalization of RAST.

C.2 Results on Additional Datasets

Apart from the mathematical reasoning and coding tasks explored in Section 3, we also experiment on more general reasoning settings. Specifically, we conduct experiments for RAST on MMLU and GPQA datasets to further demonstrate the generalization on various test settings. Similarly to previous sections, we use primary models from the Qwen-2.5 family across various sales with the corresponding RL-trained variants from SimpleRL-Zoo [76]. The decoding configurations still follow from Appendix B.

The results are shown in Table 7. We can see that the performance brought by RAST is consistent across both datasets, even if the expert RL model is still trained on mathematical reasoning datasets, which further strengthens the effectiveness and generalization of our method.

Table 7: Experiment results of RAST on general reasoning tasks with the Qwen-2.5-14B and Qwen-2.5-32B models as $\mathcal{M}_{base}$ and ΔR Simple-RL.

Models	MMLU	GPQA
Qwen-2.5-14B	62.4	24.8
$+\Delta R_{7B}$	75.1	46.7
SimpleRL-14B	77.8	50.3
Qwen-2.5-32B	61.7	38.1
$+\Delta R_{7B}$	74.8	44.7
$+\Delta R_{14B}$	79.6	49.0
SimpleRL-32B	81.4	48.3

C.3 Performance of Majority@k

We first introduce an additional metric for this additional evaluation.

Definition of Majority@k: This metric evaluates accuracy based on the majority prediction among multiple inference runs per problem. Formally,

$$\text{Majority@k} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}\left(\text{majority}\{\hat{y}_{i,1},\hat{y}_{i,2},\ldots,\hat{y}_{i,k}\} = y_i\right),$$

where the majority function returns the prediction most frequently appearing among the k inference runs for the i-th problem. Compared with **Pass@k** that represents diversity, the metric of majority@k reflects the robustness and consistency of the model performance.

We visualize the results in Figure 8, showing the progression of majority@k as k increases across six mathematical reasoning benchmarks. As expected, majority@k improves monotonically with larger k, reflecting the benefit of aggregating more diverse sampled trajectories. In most cases, our method (gray bars) bridges a significant portion of the performance gap between the base model $\mathcal{M}_{base}$ (blue line) and the RL-trained model $\mathcal{M}_{RL}$ (red line), particularly on datasets like MATH500 and GSM8K, where our approach nearly saturates the performance ceiling. This suggests strong consistency among sampled outputs and robust transfer of reasoning behaviors.

In contrast, on more challenging datasets such as AIME 2024 and AMC 2023, a wider gap remains between RAST and $\mathcal{M}_{RL}$, indicating that these tasks induce more divergent reasoning paths, where achieving high consensus remains difficult. Nevertheless, even in these harder settings, RAST offers

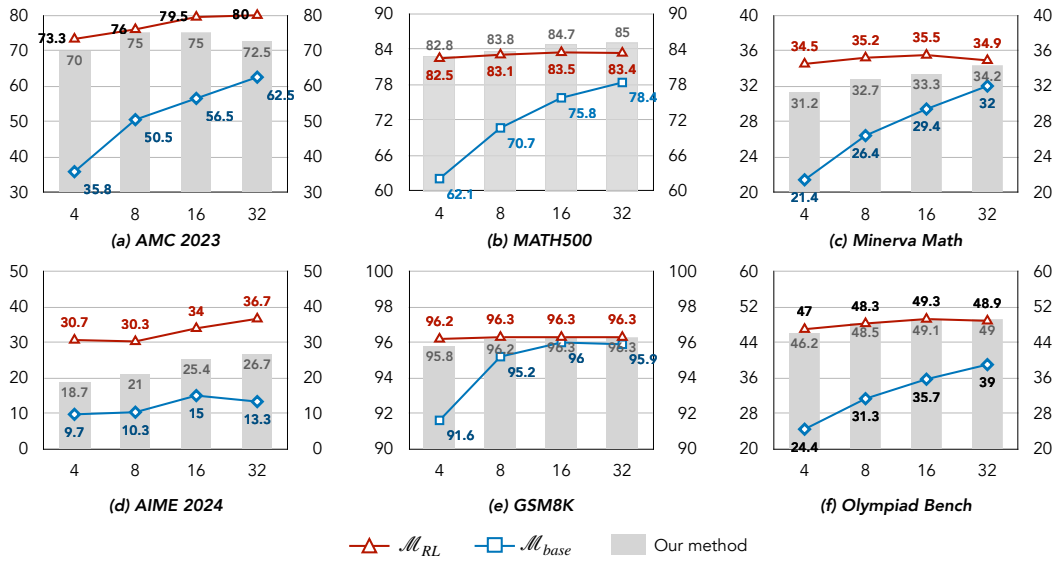


Figure 8: The illustration of majority@k for different values of k on 6 mathematical reasoning datasets, where $\mathcal{M}_{base}$ is Qwen-2.5-32B, RAST uses ΔR_{14B}, and $\mathcal{M}_{Rl}$ is the Rl-trained version of $\mathcal{M}_{base}$.

substantial improvements over the base model, demonstrating that reasoning diversity and consensus can be meaningfully enhanced without full RL training.

C.4 Response Length

To better understand the behavioral effects of RAST on model generation, we examine the average response length per problem across six mathematical reasoning benchmarks, as shown in Table 8. We report the number of output tokens generated by RAST under each setting.

Firstly, we observe that the RL-trained model generally increases the output length compared to the base models, suggesting that it encourages more verbose or exploratory reasoning paths. This behavior aligns with the goal of RL to promote step-by-step reasoning and self-verification, which naturally results in longer outputs as the model articulates intermediate steps and justifications. We find that RAST inherits this trait: applying logit deltas from RL-trained experts to base models generally leads to increased response lengths, particularly when using deltas from much smaller expert models (e.g., $\Delta R_{1.5B}$). In some cases, the response length under RAST even exceeds that of the corresponding RL-trained model. For example, on AMC and GSM8K, Qwen-2.5-7B augmented with $\Delta R_{1.5B}$ produces significantly longer outputs than both its base and RLZero counterparts, reaching an average of 1362.8 and 354.9 tokens, respectively.

Additionally, we found that the performance of each setting is generally negatively correlated with the length of the generated outputs. Specifically, the length of generated outputs is always the shortest with RAST when we apply ΔR_{14B} than $\Delta R_{1.5B}$ with 32B base models. More interestingly, we found that the best performance achieved using RAST usually comes with the shortest length, indicating that RAST can achieve both efficiency and effectiveness if configured properly. This also brings up another door for the recent topic that try to compress the reasoning trajectories of RL-tuned models [68, 77, 3, 67, 43].

Overall, these results demonstrate that RAST not only activates reasoning behaviors but also affects the generation style. This controllability opens promising directions for future work in tuning the verbosity and structure of generation by manipulating transfer signals, and supports the broader narrative that RAST serves not just as a reasoning enhancer but also as a tool for decoding-time style modulation.

https://doi.org/10.52202/085713-1663

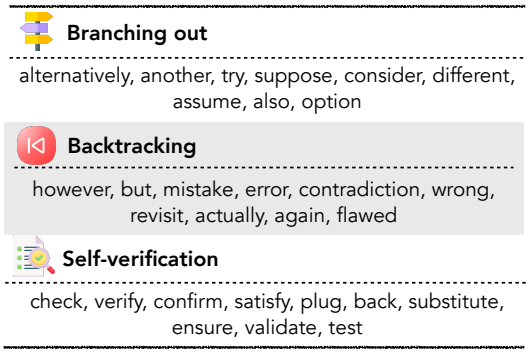


Figure 9: The set of manually curated reasoning tokens that corresponds to three key reasoning behaviours.

Table 8: The response length of output generated per problem by RAST on six mathematical reasoning benchmarks.

Models	Math500	AIME24	AMC	Minerva	Olympiad	GSM8K
Qwen-2.5-32B	566.9	1446.5	782.9	757.8	1103.4	275.6
$+\Delta R_{1.5B}$	685.1	1417.5	1024.2	799.5	978.3	333.1
$+\Delta R_{7B}$	614.5	1033.2	895.8	635.4	870.1	307.8
$+\Delta R_{14B}$	591.9	1164.6	863.9	605.7	886.9	297.6
32B-RLZero	659.2	1183.5	940.4	655.8	992.4	307.2
Qwen-2.5-14B	803.8	1138.9	820.2	921.5	1237.0	227.5
$+\Delta R_{1.5B}$	706.6	2092.0	1131.9	1070.6	1438.1	337.6
$+\Delta R_{7B}$	606.9	1092.4	919.1	636.3	889.4	309.8
14B-RLZero	682.7	1477.8	1034.9	667.1	1068.4	318.7
Qwen-2.5-7B	651.3	1765.9	756.0	657.2	1033.2	263.9
$+\Delta R_{1.5B}$	862.7	2127.4	1362.8	944.6	1123.5	354.9
7B-RLZero	682.6	1410.5	1074.6	731.7	1033.7	330.3

C.5 Case Study on Token-Level Behavior Shift

As a preliminary study mentioned in Section 2.1, we reveal that given the decoding path from RL model $\mathcal{M}_{\text{RL}}$, the base model $\mathcal{M}_{\text{base}}$ actually will largely recover the path, with PRC of 95.22% (for 32B model). In this section, we delve deeper into this study. Firstly, we repeat the study using $\tilde{\mathcal{M}}$ tuned by RAST by feeding the decoding path of $\mathcal{M}_{\text{RL}}$ to $\tilde{\mathcal{M}}$. The PRC in this experiment is $O_{\tilde{\mathcal{M}}} = 96.37\%$, which is even higher. This phenomenon indicates that RAST provides efficient guidance for the base model $\mathcal{M}_{\text{base}}$ in the search space, echoing the conclusion in [74].

Apart from the overall quantitative view, we also present a more intuitive interpretation in Figure 10. We can see that the generated output from $\mathcal{M}_{\text{base}}$ follows only one thinking and contains many erroneous steps without self-verification. However, the output from $\tilde{\mathcal{M}}$ demonstrates a markedly different reasoning behavior. It first *proposes and tests* a candidate solution, then explicitly *verifies* its correctness, and finally reasons about the function behavior to *rule out other possibilities*. To make it more rigorous, we compute the KL divergence [26] online during inference as $KLD(\mathcal{M}_{\text{base}}(t_i, x_{<i}), \tilde{\mathcal{M}}(t_i, x_{<i}))$ where $x_{<t}$ is the current prefix. Notably, we found that the KLD for tokens such as "check" reaches 837.9, which is far larger than normal tokens that usually stay below 1.0. These behavioral differences underscore the effectiveness of RAST in activating reasoning behaviors.

Math Problem:

Solve for x : $3^{2x} + 19 = 10^x$

Generated output from $\mathcal{M}_{base}$

Here's a step-by-step process to solve for x: **1.** *First, rewrite* 3^{2x} *as* $(3^2)^x$ *which equals* 9^x. **2.** *Now, rewrite the equation as* $9^x + 19 = 10^x$. **3.** *Rearrange the equation to isolate* 10^x *on one side:*$10^x = 9^x + 19$. **4.** *Divide both sides of the equation by* 9^x: $\frac{10^x}{9^x} = 1 + \frac{19}{9^x}$. **5.** *Simplify the left-hand side by using the property of exponents:*$10^x = (9 \cdot \frac{10}{9})^x = 9^x \cdot (\frac{10}{9})^x$. **6.** *Divide both sides of the equation by* 9^x: $1 = (\frac{10}{9})^x + \frac{19}{9^x}$. **7.** *Let* $y = (\frac{10}{9})^x$. *Then the equation becomes* $y = (\frac{10}{9})^x$. **8.** *Rearrange the equation to get a quadratic equation:* $y^2 - y + 19 = 0$. **9.** *Solve the quadratic equation using the quadratic formula*$y = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. *In this case,* $a = 1, b = -1$, *and* $c = 19$. ..... **15.** *Therefore,* $x = \frac{\log_{10}(\frac{1 \pm 5i\sqrt{3}}{2})}{\log_{10}(\frac{10}{9})}$.

Generated output from $\mathcal{M}_{RL}$

To solve the equation, we will proceed step by step. **1.** Isolate the exponential terms: $3^{2x} = 10^x - 19$ **2.** Analyze the equation for possible solutions: We need to find x such that both sides of the equation are equal. Let's test some simple values of x to see if we can find a solution. **3.** Test $x = 2$: $3^{2\cdot2} + 19 = 10^2$ ..... This is true, so $x = 2$ is a solution. **5.** Verify if there are other solutions: To check if there are any other solutions, we can analyze the behavior of the functions 3^{2x} and $10^x - 19$. The function $3^{2x} = (3^2)^x = 9^x$ is an exponential function with base 9. The function $10^x - 19$ is also an exponential function with base 10, shifted down by 19. Both functions are increasing, but 10^x grows faster than 9^x. For $x < 2$, 9^x is less than $10^x - 19$, and for $x > 2$, 9^x is greater than $10^x - 19$. Therefore, $x = 2$ is the only solution where the two sides of the equation are equal.

Figure 10: A case study comparing generated outputs for the same math problem sampled from MATH500 from $\mathcal{M}_{\text{base}}$ and $\tilde{\mathcal{M}}$ obtained by RAST. The red denotes erroneous thinking steps from $\mathcal{M}_{\text{base}}$ while the green texts indicate remarkably large KLD, with deeper color denoting larger KLD.

D Memory Estimation Details

We estimate the memory considering tensor parallel and CPU offloading (since these are common tricks during training) based on the following dimensions: (i) model memory footprint (FP16), (ii) optimizer states, and (iii) activations & buffers.

Model Memory Footprint. To estimate the memory consumed by model parameters, we consider GRPO training with three model instances: policy, critic, and reference. For each, the parameter size is calculated as `(model size / tensor parallelism factor) × 2 bytes`, assuming FP16 precision. For example, a 14B model with tensor parallelism of 4 would require approximately `(14B / 4) × 2B × 3 = 21GB` per GPU. This accounts for only the static model weights without any optimizer states or intermediate activations.

Optimizer States with CPU Offloading. We assume DeepSpeed ZeRO Stage 3 is used to offload optimizer states entirely to CPU memory, which is a common practice in current RL training [79]. Under the Adam optimizer, each parameter typically requires two FP32 states, leading to a total memory footprint of approximately `2× model size × 4 bytes × number of models`. These optimizer states are excluded from GPU memory but are included in CPU memory estimates. For example, in a 14B GRPO setup, this results in approximately 156 GB of CPU RAM usage for the optimizer alone. In our paper, we mainly focus on the GPU memory overheads; therefore, if using CPU offloading, the memory requirement for optimizer states could almost be neglected.

Activations and Buffer Overhead. Activation memory is estimated based on common usage patterns for transformer-based LLMs under long context lengths (e.g., 2048 tokens) and moderate batch sizes. We assume gradient checkpointing is enabled to reduce activation memory, typically resulting in 12–25 GB usage per GPU depending on model size and training configuration. Additionally, we account for 5–8 GB per GPU for auxiliary memory needs such as gradients, residual connections, attention caches, and NCCL communication buffers. These components are summed to estimate total GPU memory usage across devices.

E Future Directions

In this section, we briefly discuss the potential future directions following RAST.

Ensemble Methods. One natural extension of RAST involves ensemble strategies [9] across multiple small-scale expert models. Given that each expert model may encode slightly different

https://doi.org/10.52202/085713-1663

reasoning strategies depending on its training trajectory or initialization, aggregating their logit-level deltas could lead to more robust reasoning activation. This ensemble mechanism can be used to reduce variance, enhance generalization across tasks, and potentially adapt to unseen domains with improved resilience. We did a very initial exploration in this direction, by ensembling ΔR_{14B}, ΔR_{7B} and $\Delta R_{1.5B}$ to the 32B base model of Qwen2.5 on MATH500. We found that the results is actually around 75.6, which does not outperform RAST with a single ΔR_{14B}. We suspect the primary reason is due to the similarity reasoning behaviour encoded in ΔR across scales in the same Qwen model family. Therefore, the simple ensembling approach will hurt the performance, and also bring much memory overheads for inference. However, we argue that this is still an interesting direction for exploration if we could identify unique and diverse reasoning behaviors of different models before ensembling.

LoRA from Small Models. Another intriguing direction is to bridge the output-space corrections from RAST back into the parameter space through low-rank adaptation (LoRA) [21]. Instead of applying logit deltas directly in the decoding phase, we could naturally distil a lightweight LoRA module [80] for each RL-tuned model to imitate the adjustments suggested by smaller RL-tuned models. We may even build reasoning-centric LoRA hub [22] that enables cross-reasoning-behavior generalization and flexible reasoning pattern combination for a more controllable reasoning setting.

Beyond Reasoning. While RAST focuses on reasoning activation, the core idea of activating the reasoning behaviors from smaller RL-tuned models via output-space alignment may generalize to other capabilities, such as code generation, instruction following, or factual grounding. Future work may explore how RAST-style activation compares with or complements other alignment techniques, including preference-based fine-tuning or reward modeling.

F Limitations

While RAST demonstrates strong empirical performance and introduces a practical paradigm for decoding-time reasoning enhancement, it also comes with several limitations that suggest directions for future research.

Lack of Theoretical Understanding. Our method is largely motivated by empirical observations and intuition about how reasoning behaviors are reflected in output distributions. However, the theoretical foundations for why logit-level adjustments from small RL-tuned models can be activated effectively across model scales remain underexplored. A deeper understanding of when and why such activation works — and its potential failure modes — would help establish more rigorous guarantees and improve method design.

Limited Gains on More Difficult Benchmarks. Although RAST consistently improves performance across a range of reasoning datasets, the improvements on more challenging tasks, such as AIME, are relatively modest. These datasets often involve abstract reasoning, multi-step derivations, or domain-specific heuristics that may not be sufficiently captured by smaller expert models. This limits the degree of reasoning behavior activation and suggests that RAST may benefit from combining with more targeted adaptation techniques.

Computational Trade-offs and Expert Quality Dependency. Despite avoiding full RL fine-tuning on large models, RAST still requires access to a reasonably strong small expert model trained with RL, which can be computationally expensive to obtain. Furthermore, the quality and generalization ability of RAST are inherently tied to the effectiveness of this small expert. If the expert model is poorly aligned or fails to exhibit robust reasoning behaviors, the transferred deltas may provide little benefit.