Deep Thinking via Recursive Self-Aggregation

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Large language models (LLMs) exhibit strong reasoning capabilities through chain-of-thought (CoT) prompting, but their outputs remain unreliable due to high variability across reasoning trajectories. Parallel scaling methods like majority voting improve accuracy, but cannot "think deeper". On the other hand, sequential refinement risks locking the model into an incorrect reasoning path, from which it cannot escape. In this work, we show that LLMs can serve as aggregators over multiple CoTs, cross-referencing trajectories to identify errors and synthesize higher-quality responses. We propose Recursive Self-Aggregation (RSA), an evolutionary framework for deep thinking with increased test-time compute: aggregated CoTs are reintroduced as candidate proposals in subsequent rounds, allowing the model to progressively refine answers through iterative reasoning. This recursive aggregation, a hybrid-scaling strategy, yields monotonically improving performance with increasing token budgets. We also demonstrate that reinforcement learning (RL) finetuning can be made aggregation-aware, yielding policies that achieve superior inference-time performance under recursive aggregation compared to those trained solely for direct solution generation. On math reasoning tasks and countdown, RSA significantly outperforms baseline approaches including purely parallel and sequential strategies, with RL-trained aggregation providing additional gains.

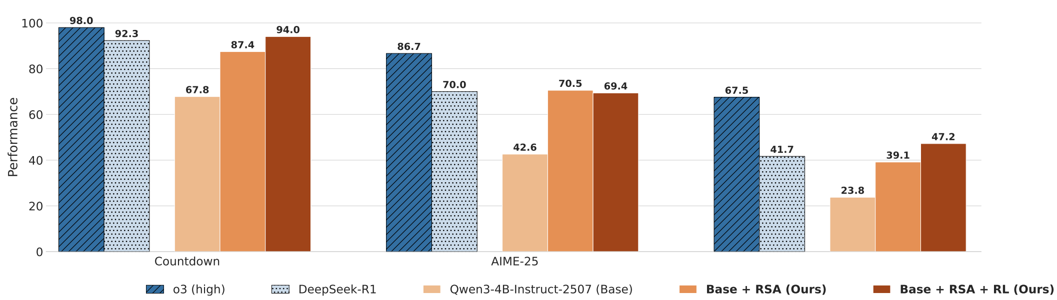


Figure 1: Performance comparison across Countdown, AIME-25, and HMMT-25. We apply Recursive Self-Aggregation (RSA) with Qwen3-4B-Instruct-2507 as a verifier-free test-time scaling procedure, bridging the gap with DeepSeek-R1. Performance is further improved through aggregation-aware RL finetuning, as described in §4.2. For RSA, we use aggregation size $K = 4$ and population $N = 16$ in all experiments, running $T = 10$ self-aggregation steps on AIME-25 and HMMT-25, and $T = 5$ steps on Countdown. Additional details are provided in Appendix A.

1 Introduction

Chain-of-thought (CoT) prompting (Wei et al., 2022) enhances the reasoning abilities of large language models (LLMs) by explicitly eliciting intermediate reasoning steps. Explicitly producing the reasoning steps allows the model to use additional compute during inference for a particular prompt, resulting in the paradigm of test-time or inference-time scaling methods (Wu et al., 2025) which enable the model to use additional compute at inference for improved performance. Common approaches include majority voting (Wang et al., 2023), which aggregates final answers across chains, and reward-based rejection sampling, which selects a single solution by filtering out low-reward solutions using a learned reward model (Cobbe et al., 2021). While these methods improve performance on some reasoning tasks, they only use the *outcomes* produced by multiple independent reasoning chains and ignore the rich information in the reasoning process itself.

Humans do not generate multiple arguments in isolation, nor do we judge them solely based on final answers. Instead, we examine intermediate steps, resolve contradictions, and combine correct partial arguments to synthesize stronger solutions. In the context of LLMs, this can be operationalized by using the reasoning capabilities of a model to aggregate and refine a set of candidate responses to produce a better solution for a given problem. This approach can be interpreted as a genetic algorithm, where each state is a population of candidate solutions that evolve through subsampling and aggregation to generate progressively better solutions.

Building on this insight, we propose *Recursive Self-Aggregation* (RSA), a test-time scaling procedure that recursively aggregates a population of reasoning chains, feeding the aggregated reasoning chains as proposals for the next round of aggregation. It is a generalization of single-sequence self-refinement to the multi-sequence setting, since the object of refinement is a set of reasoning chains rather than a single chain, allowing RSA to allocate additional compute to repeated cross-checking and refinement. This procedure has two main benefits over existing methods like rejection sampling or majority voting: (1) the model can potentially use partially correct solutions by combining correct reasoning fragments from different chains, and (2) since each set of solutions is conditioned on the previous set, the model can generate new solutions if all the candidates have incorrect reasoning. The latter aspect enables exploration of new solutions, in contrast to methods based on *i.i.d.* sampling where the final solution is restricted to the set of previously generated candidates. Empirically, even without any specific training RSA outperforms both majority voting and single-sequence self-refinement on mathematical and logical reasoning tasks, exhibiting monotonic improvements with increasing number of aggregation rounds.

As aggregating different reasoning chains can itself be seen as a reasoning task, we demonstrate that reinforcement learning (RL) post-training both for base proposals (without candidate chains) and subsequent aggregation rounds (with candidate chains) can further improve the performance of RSA. This modular design allows the aggregator to be trained independently of the proposal model. For example, it can be trained solely for aggregation using pre-generated reasoning chains from the base model, or jointly on base and aggregation prompts to improve the quality of both. Our results indicate that joint post-training across both stages leads to a marked improvement in performance compared to training for improved proposals or aggregation in isolation. As illustrated in Fig. 1, RSA with aggregation-aware RL training allows Qwen3-4B-Instruct (Yang et al., 2025) to match or exceed the performance of DeepSeek-R1 (DeepSeek-AI et al., 2025) on challenging benchmarks like Countdown, AIME-25, and HMMT-25. The simplicity and effectiveness of this approach, together with its flexibility to integrate with existing methods, highlight the potential of RSA as a general and powerful framework for test-time scaling.

2 Background

LLMs show notable performance gains when given additional test-time compute. A common way to elicit such behavior is chain-of-thought (CoT) prompting, where models are prompted to produce explicit step-by-step reasoning rather than directly generating an answer. Prompting LLMs to generate such intermediate reasoning traces has proven highly effective for complex tasks: Wei et al. (2022); Kojima et al. (2022) demonstrate that CoT prompting substantially improves arithmetic and commonsense reasoning, while Zelikman et al. (2022) show that performance on reasoning tasks can be improved by training models on their own generated reasoning chains. Building on this foundation, most test-time scaling strategies extend CoT in one of two directions. *Parallel scaling* methods leverage multiple reasoning chains generated in tandem to produce a final answer. In contrast, *sequential scaling* methods allocate test-time compute to generate a single long reasoning chain. This can involve explicit strategies to elicit self-reflection, asking the model to revisit, critique, or refine its own reasoning. *Hybrid* approaches combine these two paradigms. We refer the readers to

Zhang et al. (2025b); Snell et al. (2025) for a broad overview of such strategies and their impact on performance.

Parallel scaling. A simple but effective method is Best-of-N (Cobbe et al., 2021; Lightman et al., 2023), also called rejection sampling, which picks the best candidate from several reasoning paths using a learned verifier or reward model (*e.g.*, Zhang et al. (2025a) train generative verifiers that perform CoT reasoning for verification). Despite some progress, learning a verifier is a notoriously difficult problem (Casper et al., 2023). To address this limitation, Wang et al. (2023) proposed a verifier-free method called self-consistency, or majority voting, that samples multiple diverse CoTs and selects the answer that occurs most frequently. Self-consistency has become a strong baseline for aggregating reasoning and achieves state-of-the-art results on math benchmarks (Wang et al., 2023; Lightman et al., 2023). Our method is most closely related to parallel strategies for self-aggregation (*e.g.*, Li et al. (2025); Wang et al. (2025)) that condition an LLM on *i.i.d.* candidate reasoning chains, to produce a single refined solution. However, in contrast to these methods, RSA augments parallel aggregation with sequential scaling and recombination to iteratively improve reasoning chains.

Sequential scaling. Herel & Mikolov (2024) demonstrate the effectiveness of simple sequential scaling by inserting special "thinking" tokens after each word to use more test-time compute. Muennighoff et al. (2025) propose s1, which appends a "Wait" token at the end of each answer generation to elicit further reasoning and self-reflection. Another option is to let the model self-evaluate and refine its own reasoning chains. In this direction, Madaan et al. (2023) introduce Self-Refine which uses the same model to iteratively provide feedback on the generated answer and then refine the answer based on the feedback. Related approaches explicitly prompt models to "reflect" or critique their solutions to reduce reasoning errors, *e.g.*, SCoRe (Kumar et al., 2024) and PAG (Jiang et al., 2025). Saunders et al. (2022) and Perez et al. (2023) found that eliciting self-critiques from a model can catch factual and logical errors, while Shinn et al. (2023) proposed Reflexion, where agents use verbal self-reflection as memory to iteratively improve performance in interactive environments. Other methods such as Chain-of-Verification (Cove; Dhuliawala et al., 2024) plan verification questions to fact-check draft responses before finalizing them. Aggarwal & Welleck (2025) train models with RL to reason for a given thinking budget. Strong reasoning models trained directly with RL already exhibit self-verification and refinement in their CoTs through emergent learned skills (DeepSeek-AI et al., 2025), but their performance can be further enhanced through the explicit elicitation strategies outlined above.

Hybrid scaling. While parallel strategies can produce more diverse reasoning chains relative to sequential strategies, they lack self-refinement and self-improvement ability of sequential approaches. This paves the way for hybrid strategies that can get the best of both worlds by exploring diverse chains in parallel while selectively refining them sequentially. Tree of thoughts (ToT; Yao et al., 2023) combines parallel and sequential generation by exploring reasoning paths structured as a search tree with partial feedback. Hao et al. (2023) propose using Monte Carlo tree search (MCTS; Kocsis & Szepesvári, 2006) instead of standard tree search approaches in the context of test-time scaling. Wang et al. (2024) introduces "Mixture-of-Agents" (MoA) which combines parallel scaling using multiple LLMs and leverages another LLM for sequential refinement of these multiple responses. Our method, Recursive Self-Aggregation (RSA), falls into this hybrid category, integrating both parallel exploration and sequential self-improvement. RSA is closely related to MoA; whereas MoA preserves diversity in candidate proposals by ensembling multiple LLMs, our method relies on a single model and achieves diversity by maintaining a population size larger than each aggregation batch size, a factor we identify as critical in §5.3. This approach of maintaining a constantly evolving population of solutions is inspired by genetic algorithms, and by using a single model we are leveraging its *own* aggregation abilities rather than relying on multiple models or external verifiers. Our method is also related to other evolutionary LLM approaches (*e.g.*, Novikov et al. (2025); Lee et al. (2025)), but unlike these, it does not rely on external verifiers or explicit fitness functions, making it more broadly applicable.

3 Evolving thoughts through recursive self-aggregation

In this section, we describe our approach, *Recursive Self-Aggregation* (RSA), that is designed to improve both the quality and accuracy of model responses by effectively utilizing test-time compute. RSA frames reasoning as an evolutionary process, where candidate reasoning chains are iteratively refined through self-aggregation, inspired by the crossover and mutation steps in a genetic

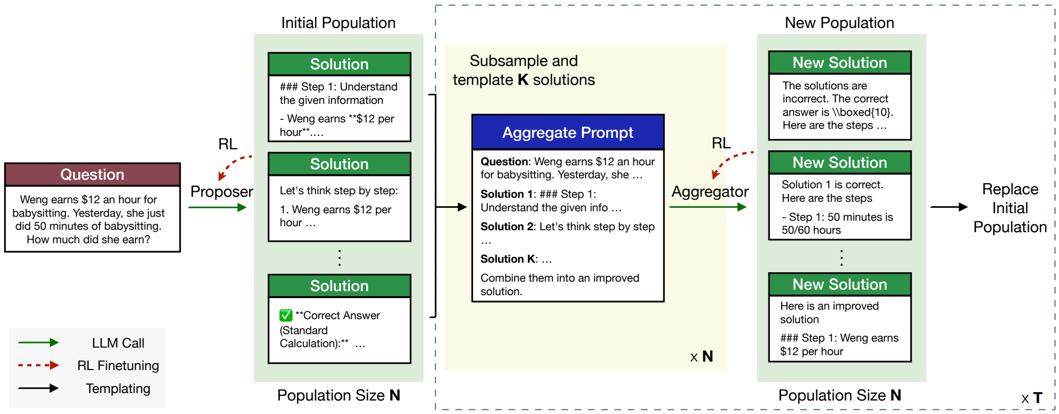


Figure 2: We propose *Recursive Self-Aggregation* (RSA), a hybrid framework for test-time scaling which generates a population of N solutions for a given prompt and then recursively updates them over T steps. Each update step relies on subsampling K distinct solutions from the current population and leveraging a LLM call to provide an improved solution. Optionally, RSA can be further enhanced by post-training the LLM for improved proposal or aggregation generation, or both.

algorithm. RSA generalizes single-trajectory refinement methods by jointly considering a diverse set of trajectories to refine at each step, which enables the model to identify correct partial reasoning steps and discard erroneous steps in the candidate solutions. Fig. 2 illustrates the core components of RSA, which we describe in this section.

Population of trajectories. RSA maintains a population $\mathcal{P}_t = \{\tau_1^{(t)}, \dots, \tau_N^{(t)}\}$ of N candidate CoTs. At the first step, these N candidates are sampled independently from the same model using a standard prompt that elicits chain-of-thought reasoning.

Aggregation of trajectories. At each step t, we form N aggregation sets:

$$\mathcal{S}_t = \{S_1^{(t)}, S_2^{(t)}, \dots, S_N^{(t)}\}, \quad S_i^{(t)} \subseteq \mathcal{P}_t, \; |S_i^{(t)}| = K,$$

where each aggregation set contains K candidates sampled uniformly without replacement from the population $\mathcal{P}_t$. An aggregate prompt formed from each aggregation set $S_i^{(t)}$ directs the model to integrate the K trajectories into a refined solution, see Fig. 2. Note that trajectories need not terminate in a complete answer; even partial CoTs can provide valuable signal during aggregation. The N aggregation sets yield N refined responses, that form the population for the next generation:

$$\tau_i^{(t+1)} \sim p_\theta(\cdot \mid S_i^{(t)}), \quad \mathcal{P}_{t+1} = \{\tau_1^{(t+1)}, \dots, \tau_N^{(t+1)}\}.$$

The choice of K depends on how many distinct trajectories can fit within the model's context window. Note that $K = 1$ is equivalent to single-trajectory self-refinement. In practice, we find that even setting $K = 2$ offers significant improvement over $K = 1$, highlighting that using diverse solutions for aggregation is a key factor.

Since all reasoning chains come from the same LLM, self-aggregation can quickly lead to loss of diversity due to excessive reuse of reasoning patterns that occur in many trajectories in the population. Maintaining a population size N that is large relative to the aggregation size K helps ensure sufficient variability for recombination and thus prevent such collapse. However, a very large N relative to K can slow the convergence of the population as a whole: high-quality patterns require more aggregation rounds to dominate the batch. We explore these tradeoffs in §5.3.

Recursive self-aggregation. The process described above is iterated for a fixed number of steps T, and at each step the population $\mathcal{P}_t$ is transformed into $\mathcal{P}_{t+1}$ via the aggregation and resampling process described above. This sequential loop allows scaling with more test-time compute: errors and inconsistencies are gradually pruned away implicitly during aggregation, while favorable reasoning patterns are reinforced through repeated recombination. Consequently, we expect overall diversity within the population to generally decrease as t increases, accompanied by a monotonic improvement in success rate. The final answer is obtained at step T, either by sampling a response uniformly at random from $\mathcal{P}_T$ or by applying an oracle-free aggregation strategy, such as majority vote over the final answers in the population.

4 Training aggregators with reinforcement learning

RSA is an effective test-time scaling approach for improving the reasoning capabilities of any pre-trained language model. In this section we discuss how training the model specifically for aggregation using reinforcement learning (RL) can provide further performance gains.

4.1 Reinforcement learning for LLM reasoning

In standard RL post-training, a base LLM $p_{\theta_{\text{base}}}$ is trained to generate chains of thought (CoTs) $\mathbf{z}$ conditioned on a prompt $\mathbf{x}$ to maximize the reward R, quantified typically by either a learned model trained on human feedback (Ouyang et al., 2022) or from an oracle verifier (Luong et al., 2024). We focus on verifiable math and reasoning tasks, where solution correctness can be exactly determined by an oracle. Given a question-answer pair $(\mathbf{x}, \mathbf{y})$, a CoT is deemed correct if it produces the answer $\mathbf{y}$. For example, if a model is prompted to enclose its answer in a `\boxed{}` command, the reward is

$$R(\mathbf{z}, \mathbf{y}) = \mathbb{1}_{\text{Extract}(\mathbf{z})=\mathbf{y}}, \tag{1}$$

where Extract$(\cdot)$ denotes the function that returns the boxed string. Given a dataset of question-answer pairs $\mathcal{D}$, the RL finetuning objective is

$$\theta_{\text{post}} = \arg\max_{\theta} \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{D}} \mathbb{E}_{\mathbf{z}\sim p_\theta(\cdot|\mathbf{x})} \left[R(\mathbf{z}, \mathbf{y}) - \beta \mathbb{KL}\left[p_\theta(\cdot|\mathbf{x}) \,\|\, p_{\theta_{\text{base}}}(\cdot|\mathbf{x})\right]\right] \tag{2}$$

where β is a hyperparameter that controls KL divergence of the LLM policy from the base model. This objective can be optimized using standard online RL algorithms such as REINFORCE (Ahmadian et al., 2024) or GRPO (Shao et al., 2024).

4.2 Aggregator training

While standard RL post-training can improve reasoning, its benefits are limited when paired with inference-time strategies, as they rely solely on (a) individual reasoning chains in isolation, and (b) signals of correctness that ignore the complexity and subtlety present in entire reasoning chains. To address this limitation, we propose framing aggregation as a reasoning task: we finetune a language model that takes a set of reasoning chains (proposals) as input and is tasked with aggregating information from them to provide improved reasoning:

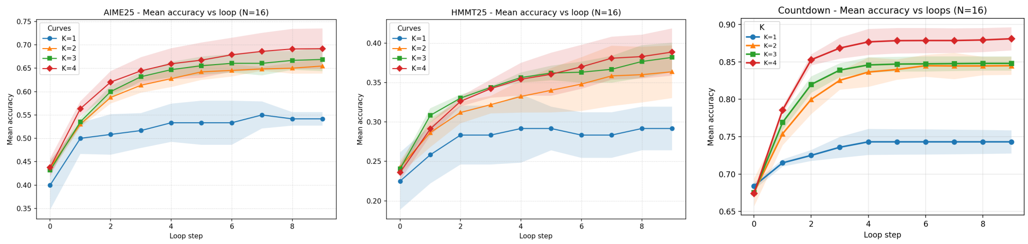
$$\max_{\theta} \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{D}} \mathbb{E}_{\mathbf{z}_{1:K}\sim p_{\theta_{\text{prop}}}(\cdot|\mathbf{x})} \left[\mathbb{E}_{\mathbf{z}\sim p_\theta(\mathbf{z}|\mathbf{x},\mathbf{z}_{1:K})} \left[R(\mathbf{z}, \mathbf{y})\right] - \beta \mathbb{KL}\left[p_\theta(\cdot|\mathbf{x}, \mathbf{z}_{1:K}) || p_{\theta_{\text{base}}}(\cdot|\mathbf{x}, \mathbf{z}_{1:K})\right]\right] \tag{3}$$

where K chains of thought $\mathbf{z}_{1:K}$ are sampled *i.i.d.* from a proposal model $p_{\theta_{\text{prop}}}$. In our experiments, we either consider the base model $p_{\theta_{\text{base}}}$ or the RL post-trained model $p_{\theta_{\text{post}}}$ from Equation 2 as the proposal generator.

Joint training. Equation (3) optimizes for improvement in subsequent rounds of aggregation over simply using θ_{base} for aggregation. However, the base performance, at the initial step without aggregation, of this post-trained model still lags that of θ_{post} which is fine-tuned to improve proposal performance. To alleviate this, we consider a simple setup where the finetuning is done with a mix of (2) and (3), which allows training of the same model p_θ for improving both aggregation as well as initial proposals.

5 Experiments

We begin by analyzing how key algorithmic parameters – the population size N, the number of iterations T, and the aggregation set size K – affect performance. Sequential scaling is primarily dependent on the number of loop iterations T (§5.1), which increases thinking time linearly. Parallel scaling, on the other hand, is controlled by two parameters: the aggregation set size $|\mathcal{S}_i^{(t)}| = K$ (§5.2) and the overall population size $|\mathcal{P}_t| = N$. To shed light on the relationship between these parameters and their impact on task performance, we analyze population statistics in §5.3. For our experiments, we consider math problems from the AIME-25 and HMMT-25 datasets (Balunović et al., 2025), and a logical reasoning task in the form of the game of Countdown (Gandhi et al., 2024). We use Qwen3-4B-Instruct-2507 (Yang et al., 2025) across experiments, unless otherwise stated, since it exhibits strong baseline performance on the evaluated tasks without finetuning, but does not fully saturate them, leaving room for improvement through test-time scaling. We benchmark performance against stronger reasoning models and test time scaling strategies in §5.5. Finally, in §5.6 we demonstrate the effectiveness of the aggregation-aware RL training introduced in §4.2.



(a) AIME-25 (b) HMMT-25 (c) Countdown

Figure 3: Accuracy as a function of loop step for different values of aggregation size K, for fixed population size $N = 16$. Error bands indicate standard deviation over 4 seeds. All results were obtained using the base models without additional RL finetuning.

5.1 RSA yields monotonic improvement over iterations

Effective sequential scaling is characterized by monotonic improvements with increased thinking time. As illustrated in Fig. 3, RSA consistently improves the Mean@N performance with increased thinking time in all settings. We report the population accuracy for $T = 10$ loop iterations. Here, the first step uses the proposal model, while the remaining 9 steps correspond to successive rounds of aggregation. We observe that the accuracy improves sharply in the first few rounds and continues to improve with more steps.

5.2 Increasing aggregation size K improves performance

As shown in Fig. 3, increasing the aggregation size K for a fixed $N = 16$ consistently improves the Mean@N performance, with the largest gain observed when moving from $K = 1$ to $K = 2$. For $K = 1$, trajectories can be improved only through self-refinement, without any mixing across the population. This highlights the clear benefit of aggregation over self-refinement alone. However, scaling K beyond a handful of trajectories is challenging due to the rapid growth of context length. Details about the context length used for each task are provided in Appendix A.

5.3 Pass@N as a predictor of asymptotic performance

We now study the impact of population size N on performance. Recall that N denotes the total population size, or the number of unique candidates available for sampling at each step. Therefore, N must be at least as large as the aggregation size K. For the following analysis, we consider the Pass@N metric, which relies on access to an oracle verifier. While such access is not typically assumed, it is useful here for studying asymptotic performance. Recall that the Pass@N score for a batch of N answers is defined as 1.0 if at least one candidate is correct according to the oracle verifier.

The left column of Fig. 4 reports Pass@N across loop iterations for different values of N. As expected, larger N yields higher Pass@N scores. Importantly, the Pass@N score remains stable across iterations, which suggests that the strongest candidate in each batch is typically preserved throughout the evolutionary process. The middle column shows that increasing N generally improves performance across different values of K owing to more diverse reasoning chains within the population. Finally, the right column plots the gap between Pass@N and Mean@N, which steadily declines over iterations, indicating convergence of the batch toward uniformly high-quality candidates.

The observation that Mean@N steadily approaches Pass@N on some tasks is remarkable, as it shows that RSA allows to achieve the Pass@N score *without relying on an oracle at inference time*. This, combined with the evidence presented in the middle column of Fig. 4, implies that N should be scaled as large as possible, as it improves the Pass@N score to which the evolutionary procedure is expected to asymptotically converge. However, to ensure adequate mixing within a reasonable number of iterations T, K must also be increased in tandem. As shown in the right column, larger K accelerates the convergence of Mean@N toward Pass@N.

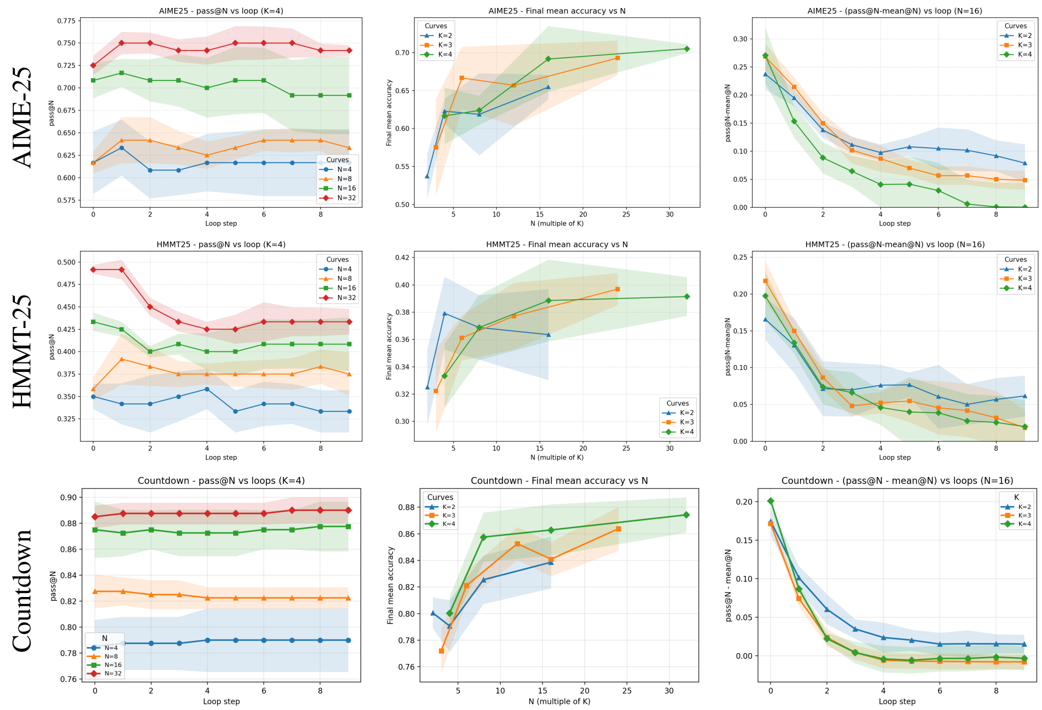


Figure 4: Ablation studies over N and K, to analyze their impact on asymptotic performance. **Left:** Pass@N across loop iterations for varying values of N. **Middle:** Final accuracy at loop iteration 10 as a function of N for different aggregation sizes K, with N chosen as $K \cdot 2^i$. **Right:** The difference between Pass@N and Mean@N across loop iterations for different values of K.

5.4 Tradeoffs

While our results indicate that increasing N, K (to a certain extent for a fixed N), and T improve performance, the key question is how to scale them relative to one another given a limited compute budget. When long sequential reasoning is feasible, a large T can be used, which allows for smaller K provided that N is large, since N primarily governs asymptotic performance, whereas K mainly controls the convergence rate. Conversely, when T is limited due to time constraints and increasing K is impractical (*e.g.*, due to context length constraints), N should also be reduced; a large population that fails to mix effectively is less useful than a smaller batch that evolves together more rapidly.

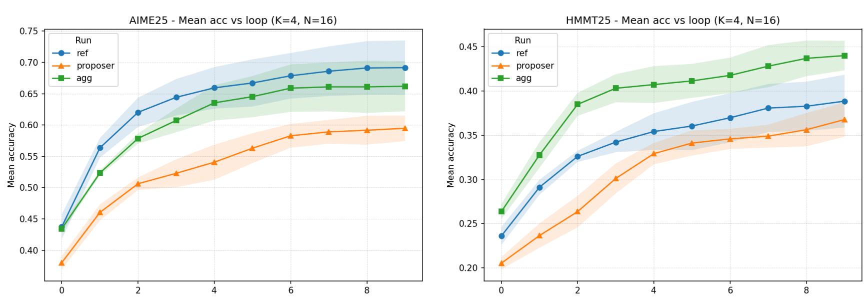
5.5 Evaluating against baselines

In Table 1, we report the performance of RSA with $K = 4$ and $N = 16$. Across all tasks, we evaluate the base (zero-shot) performance of Qwen3-4B-Instruct-2507. To benchmark against pure sequential scaling, we report results for single-trajectory self-refinement, corresponding to the $K = 1$ setting of RSA. We further compare to a strong sequential scaling baseline: s1's "Wait" budget-forcing strategy (Muennighoff et al., 2025). In this approach, the token "Wait" is appended whenever the model produces a final answer, prompting continued reasoning and self-reflection—capabilities reinforced through finetuning on a curated dataset. For a fair comparison, we evaluate the publicly released model s1-32B and reference their reported performance under $4\times$ budget forcing (which typically saturates the model's 32k context window). For our approach, we run RSA on Qwen3-4B-Instruct-2507 for $T = 10$ iterations on all settings including AIME, HMMT, and DeepScaleR, and Countdown. For the s1 experiments, we report the AIME-2025 results from their paper alongside RSA runs with $T = 5$. The majority voting baseline is budget-matched to RSA in terms of total generations, using $N \times T$ trajectories per question under identical response length constraints. To further highlight the effectiveness of our method, we also report the zero-shot evaluation results from reasoning models substantially stronger than Qwen3-4B-Instruct-2507 in the absence of deep test-time thinking, specifically DeepSeek-R1 (DeepSeek-AI et al., 2025) and o3-mini (high).

Table 1: We report mean scores of RSA and other baselines across four tasks. RSA results are for $K = 4, N = 16$. For AIME, HMMT, and Countdown with Qwen, we set $T = 10$. For all s1 evals we use $T = 5$. Same T is used for RSA and self-refinement. Majority vote for each task uses the corresponding $N \times T$ candidates, budget-matching RSA.

Model	**Method**	**AIME-25**	**HMMT-25**	**DeepScaleR**	**Countdown**
DeepSeek-R1	Zero-shot	70.0	41.67	–	92.3
o3-mini (high)	Zero-shot	86.67	67.50	–	98.0
Qwen3-4B-Instruct-2507	Zero-shot	42.60	23.75	48.39	67.41
	Self-refinement	54.17	29.17	54.49	74.31
	Majority vote	66.67	33.33	55.67	75.29
	RSA	69.17	38.85	56.04	88.09
s1-32B	Zero-shot	27.91	–	–	–
	Self-refinement	33.33	–	–	–
	“Wait” 1x	30.0	–	–	–
	“Wait” 4x	36.7	–	–	–
	Majority vote	43.33	–	–	–
	RSA	41.88	–	–	–

Remarkably, RSA with Qwen3-4B-Instruct-2507 almost entirely bridges the performance gap with DeepSeek-R1, a 671B-parameter reasoning model. The result improves further under the $N = 32$ configuration, as reported in Fig. 1. It also consistently outperforms majority voting with the same generation budget, trading off total parallelism for a hybrid approach. With s1-32B, RSA achieves a substantial improvement over the “Wait” budget-forcing method on AIME-25. Interestingly, majority voting with $16 \times 5 = 80$ generations attains the strongest performance in this setting. Notably, Muennighoff et al. (2025) did not include a comparison to majority voting. We observe that the s1 model generally prefers longer generations, but its 32,000-token context limit required us to enforce a cutoff length of 4,096 tokens for all RSA responses. This restriction might explain why the s1 task is the only one where RSA is beaten by majority voting.



(a) AIME-25

(b) HMMT-25

Figure 5: We plot the Mean@N score for different tasks across iterations of Recursive Self-Aggregation performed for $T = 10$ iterations on RL finetuned policies. Error bands indicate standard deviation across 4 seeds.

5.6 Aggregation-aware RL training boosts performance

We examine the effect of RL finetuning on aggregation performance, particularly whether aggregation-aware training leads to stronger performance when RSA is applied at inference time.

Setup. We perform experiments on math problems, where we randomly sample $36,000$ problems from the DeepScaleR dataset (Luo et al., 2025), a high-quality collection of challenging math problems that often require long CoT reasoning. We evaluate on unseen math problems from the AIME-25 and HMMT-25 datasets (Balunović et al., 2025).

We use Qwen3-4B-Instruct-2507 as the reference policy and train the *aggregator* policy as described in §4.2. Specifically, we first sample $K = 4$ proposals from the reference policy and augment the dataset with aggregation prompts containing these candidates (see Appendix A for the exact prompts used). We also include the standard prompts containing only the question in this augmented dataset. As a baseline, we train a *proposer* policy via standard RL finetuning, as described in §4.1, to predict answers without conditioning on candidate proposals. We call it the *proposer* since it is only trained to generate the initial population of candidates at the first step of the RSA procedure. We train both policies for 140 steps with RLOO (Ahmadian et al., 2024). Other hyperparameter details are listed in Appendix A. During evaluation, we run RSA for $T = 10$ iterations.

Results. Fig. 5 reports the results of this experiment. We plot the Mean@N score for each task, across RSA iterations. The Mean@N consistently improves throughout the evolutionary process, for all three policies (reference, *proposer*, and *aggregator*). The *proposer* policy, trained only on the base prompts from DeepScaleR consistently performs worse than the *aggregator* across both the tasks. Interestingly, RL finetuning does not necessarily outperform the reference policy when using RSA. The reference policy outperforms the *proposer* policy in the case of HMMT-25, and even the aggregator policy in the case of AIME-25. Due to compute constraints, we trained with maximum response length 4096 to prevent the aggregator context from becoming prohibitively large, but increased it to 8192 tokens during evaluation. This might explain why the *aggregator* policy performs slightly worse than the reference policy on AIME. We plan to investigate the effect of aggregator-aware RL training further in future work.

6 Conclusion

Recursive Self-Aggregation generalizes both single-trajectory self-refinement and fully parallel majority-vote aggregation into a unified test-time scaling framework that achieves state-of-the-art performance in practice. Its predictable trade-offs across scaling parameters make it a powerful and easy-to-implement strategy for reliably enhancing LLM reasoning through test-time thinking.

Future work. More extensive ablations with RSA across model scales and a broader suite of tasks could provide deeper insights into its applicability. In this work, we adopted a simple approach to augment the RL dataset by adding aggregation prompts, but this training setup still doesn't optimize for the multi-step test-time procedure. A promising direction for future work is multi-step RL training of recursive aggregation policies, which could, for example, encourage greater diversity in the intermediate candidate solutions. This stands in contrast to the current approach, which greedily optimizes for prediction the final answer at each step and may inadvertently reduce population diversity.

References

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

Ahmadian, A., Cremer, C., Gallé, M., Fadaee, M., Kreutzer, J., Pietquin, O., Üstün, A., and Hooker, S. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024. URL `https://arxiv.org/abs/2402.14740`.

Balunović, M., Dekoninck, J., Petrov, I., Jovanović, N., and Vechev, M. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL `https://matharena.ai/`.

Casper, S., Davies, X., Shi, C., Gilbert, T. K., Scheurer, J., Rando, J., Freedman, R., Korbak, T., Lindner, D., Freire, P., et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

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

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

Dhuliawala, S. Z., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., and Weston, J. E. Chain-of-verification reduces hallucination in large language models, 2024. URL `https://openreview.net/forum?id=VP20ZB6DHL`.

Gandhi, K., Lee, D. H., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. Stream of search (sos): Learning to search in language. In *First Conference on Language Modeling*, 2024.

Hao, S., Gu, Y., Ma, H., Hong, J., Wang, Z., Wang, D., and Hu, Z. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 8154–8173, 2023.

Herel, D. and Mikolov, T. Thinking tokens for language modeling. *arXiv preprint arXiv:2405.08644*, 2024.

Jiang, Y., Xiong, Y., Yuan, Y., Xin, C., Xu, W., Yue, Y., Zhao, Q., and Yan, L. Pag: Multi-turn reinforced llm self-correction with policy as generative verifier. *arXiv preprint arXiv:2506.10406*, 2025.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

Kumar, A., Zhuang, V., Agarwal, R., Su, Y., Co-Reyes, J. D., Singh, A., Baumli, K., Iqbal, S., Bishop, C., Roelofs, R., et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Lee, K.-H., Fischer, I., Wu, Y.-H., Marwood, D., Baluja, S., Schuurmans, D., and Chen, X. Evolving deeper llm thinking, 2025. URL `https://arxiv.org/abs/2501.09891`.

Li, Z., Feng, X., Cai, Y., Zhang, Z., Liu, T., Liang, C., Chen, W., Wang, H., and Zhao, T. Llms can generate a better answer by aggregating their own responses. *arXiv preprint arXiv:2503.04104*, 2025.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

Luo, M., Tan, S., Wong, J., Shi, X., Tang, W. Y., Roongta, M., Cai, C., Luo, J., Li, L. E., Popa, R. A., and Stoica, I. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

Luong, T. Q., Zhang, X., Jie, Z., Sun, P., Jin, X., and Li, H. Reft: Reasoning with reinforced fine-tuning. *arXiv preprint arXiv:2401.08967*, 2024.

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

Muennighoff, N., Yang, Z., Shi, W., Li, X. L., Fei-Fei, L., Hajishirzi, H., Zettlemoyer, L., Liang, P., Candès, E., and Hashimoto, T. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J., Mehrabian, A., et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Perez, E., Ringer, S., Lukosiute, K., Nguyen, K., Chen, E., Heiner, S., Pettit, C., Olsson, C., Kundu, S., Kadavath, S., Jones, A., Chen, A., Mann, B., Israel, B., Seethor, B., McKinnon, C., Olah, C., Yan, D., Amodei, D., Amodei, D., Drain, D., Li, D., Tran-Johnson, E., Khundadze, G., Kernion, J., Landis, J., Kerr, J., Mueller, J., Hyun, J., Landau, J., Ndousse, K., Goldberg, L., Lovitt, L., Lucas, M., Sellitto, M., Zhang, M., Kingsland, N., Elhage, N., Joseph, N., Mercado, N., DasSarma, N., Rausch, O., Larson, R., McCandlish, S., Johnston, S., Kravec, S., El Showk, S., Lanham, T., Telleen-Lawton, T., Brown, T., Henighan, T., Hume, T., Bai, Y., Hatfield-Dodds, Z., Clark, J., Bowman, S. R., Askell, A., Grosse, R., Hernandez, D., Ganguli, D., Hubinger, E., Schiefer, N., and Kaplan, J. Discovering language model behaviors with model-written evaluations. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13387–13434, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.847. URL https://aclanthology.org/2023.findings-acl.847/.

Saunders, W., Yeh, C., Wu, J., Bills, S., Ouyang, L., Ward, J., and Leike, J. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

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

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

Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36:8634–8652, 2023.

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

Stojanovski, Z., Stanley, O., Sharratt, J., Jones, R., Adefioye, A., Kaddour, J., and Köpf, A. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards, 2025. URL https://arxiv.org/abs/2505.24760.

Wang, J., Wang, J., Athiwaratkun, B., Zhang, C., and Zou, J. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.

Wang, Q., Zhao, P., Huang, S., Yang, F., Wang, L., Wei, F., Lin, Q., Rajmohan, S., and Zhang, D. Learning to refine: Self-refinement of parallel reasoning in llms, 2025. URL https://arxiv.org/abs/2509.00084.

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

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wu, Y., Sun, Z., Li, S., Welleck, S., and Yang, Y. Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., and Narasimhan, K. R. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=5Xc1ecx01h.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. STar: Bootstrapping reasoning with reasoning. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_3ELRdg2sgI.

Zhang, L., Hosseini, A., Bansal, H., Kazemi, M., Kumar, A., and Agarwal, R. Generative verifiers: Reward modeling as next-token prediction. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Zhang, Q., Lyu, F., Sun, Z., Wang, L., Zhang, W., Hua, W., Wu, H., Guo, Z., Wang, Y., Muennighoff, N., King, I., Liu, X., and Ma, C. A survey on test-time scaling in large language models: What, how, where, and how well?, 2025b. URL https://arxiv.org/abs/2503.24235.

A Experiment Details

Inference settings For LLM inference, we use VLLM (Kwon et al., 2023) for both RSA and baselines like majority voting. We keep sampling parameters consistent across experiments. We set the sampling temperature = 1.0, top_p = 1.0, and min_p = 1.0 (all default settings). For AIME-25, HMMT-25 and Countdown evals, we set the maximum response length to 8192, and for DeepScaleR we set it to 4096.

Evaluation info For the DeepScaleR metrics in Table 1, we report results from a single seed evaluated on 512 *i.i.d.* samples from the dataset. For Countdown, we generate 100 random problems using the default reasoning-gym schema and compute the average reward over 4 random seeds. We use 4 random seeds when evaluating AIME-25 and HMMT-25, each of which consists of 30 problems.

The reported AIME and HMMT zero-shot scores for DeepSeek-R1 and o3-mini (high) in Table 1 are taken from MathArena, while the Countdown scores are taken from Stojanovski et al. (2025). For s1, we report the "Wait" Nx scores on AIME-25 from their paper (Muennighoff et al., 2025), while the remaining metrics were collected by us using their released model.

RL experiments We use `verl` (Sheng et al., 2024) to train the *aggregator* and *proposer* policies described in §5.6 on 36,000 samples from the DeepScaleR dataset. Training is performed with the RLOO (Ahmadian et al., 2024), using a learning rate of 2×10^{-6}, KL penalty in the reward with $\beta = 0.01$, batch size $= 256$, and 140 training steps (one epoch). The maximum response length is set to 4096 during training, but increased to 8192 during evaluation on AIME and HMMT. We note that training with the shorter sequence length is due to compute constraints, which may explain the slightly worse performance of the *aggregator* compared to the reference policy on AIME. All other training parameters are set to the verl defaults.

LLM prompts For step 1 of RSA (initial proposal generation), we use the following prompt:

> [Question]
> Let's think step by step and output the final answer within [Answer_format].

Where [Answer_format] is \\boxed{ } for Math tasks and <answer></answer> for Countdown. We use the following prompt for the subsequent aggregation steps of RSA, with K aggregation size:

> You are given a problem and several candidate solutions. Some candidates may be incorrect or contain errors. Aggregate the useful ideas and produce a single, high-quality solution. Reason carefully; if candidates disagree, choose the correct path. If all are incorrect, then attempt a different strategy. End with the final result in [Answer_format].
> Problem:
> [Question]
> Candidate solutions (may contain mistakes):
> —- Solution 1 —-
> [Proposal_1]
> —- Solution 2 —-
> [Proposal_2]
> ⋮
> —- Solution [K] —-
> [Proposal_K]
> Now write a single improved solution. Provide clear reasoning and end with the final answer in [Answer_format].

For the self-refinement experiments ($K = 1$), we use a slight variation of the aggregation prompt:

You are given a problem and a candidate solution. The candidate may be incomplete or contain errors. Refine this trajectory and produce an improved, higher-quality solution. If it is entirely wrong, attempt a new strategy. End with the final result in [Answer_format].
Problem:
[Question]
Candidate solution (may contain mistakes):
—- Candidate —-
[Proposal]
Now refine the candidate to an improved solution. Provide clear reasoning and end with the final answer in [Answer_format].

B Limitations

The primary limitation of RSA is the rapid growth of context length with increasing aggregation size K. While larger K generally improves performance, it also increases memory requirements, especially since N must scale accordingly to realize these gains. Moreover, model performance often degrades at extremely long context lengths, implying the existence of a threshold K beyond which performance declines. A second limitation is efficiency: unlike purely parallel strategies such as majority voting, the recursive self-aggregation in RSA introduces a sequential component that can be slow, making it less suitable for scenarios requiring fast responses.

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: Experiments in §5.5.

   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: See Appendix B

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: No 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: See Appendix A

   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: [No]

   Justification: Code will be released in the future with the archival version.

   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).

- 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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: See Appendix A

   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: Std over 4 seeds reported for all plots in §5. Only mean over 4 seeds is reported for scores in Table 1, and details provided in Appendix A

   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: [No]

   Justification: Will be released with archival version

   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: The research conforms to 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: This work has no specific broader impact beyond the usual considerations associated with improving the capabilities of large language models.

   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]

   Justification: No model or data released with the paper.

   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: All models and code frameworks used are credited.

    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: [NA]

    Justification: No new assets introduced.

    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: No crowdsourcing or human subjects used.

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: Does not involve crowdsourcing or using 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: LLMs only used for editing and formatting purposes.

    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.