Consolidating Reasoning with Test-Time Training

Haotian Wu* **Zeming Chen*** **Hao Zhao** **Antoine Bosselut**

EPFL

Abstract

Effective multi-agent collaboration requires finalizing solutions from reasoning that is distributed across the individual attempts of many agents. However, useful intermediate reasoning can remain unexploited when collaboration is limited to voting on conclusions reached among individual agents, or aggregating long, noisy collections of reasoning trajectories in context. In this work, we propose **TTT-Consolidate**, a framework that uses multi-agent reasoning trajectories as adaptation data to improve a model's subsequent reasoning. TTT-Consolidate jointly consolidates trajectories into parametric representations through test-time training, then generates a new solution by conditioning on the consolidated information. We meta-learn *how to consolidate* using bi-level optimization at training time, where an inner loop encodes reasoning trajectories into a lightweight LoRA adapter, and an outer loop learns meta-parameters for the inner loop such that the consolidation is helpful for the model to synthesize a final solution.

As no large-scale corpus of diverse reasoning trajectories exists for training aggregation methods, we construct the *OpenParallelThinking* corpus composed of 12,810 math problems paired with a mixture of trajectories from six LLMs. Evaluated on challenging math and STEM benchmarks, TTT-Consolidate outperforms the best aggregation baselines by up to 20 percentage points, even under matched compute and runtime settings, with the gap widening at larger model scales. TTT-Consolidate also recovers from unsuccessful individual trajectories more robustly, an ability that improves as the number of available (even incorrect) trajectories increases. Finally, we demonstrate that TTT-Consolidate benefits from both SFT and RL-based outer loop objectives. We will release TTT-Consolidate and *OpenParallelThinking* for use in future work on reasoning aggregation.

1 Introduction

Multi-agent systems offer a promising way to extend the reasoning capabilities of large language models by allowing models to exchange, critique, and build on one another's responses. Early work demonstrates benefits from multi-agent debate and layered synthesis of outputs from different models [18, 58]. More recent work explores how learned teamwork strategies can combine partial reasoning into solutions absent from members' independent answers, and how communicating agents can build on intermediate discoveries while solving problems [42, 43]. Together, these directions motivate an important question: *Can a model learn from other agents' reasoning to construct a better solution?*

Effective collaboration requires aggregating distributed partial reasoning into a better solution by exploiting useful intermediate reasoning even when the attempts reach incorrect conclusions. However, common trajectory aggregation methods, such as majority voting, reward-guided selection, and best-of-N sampling [7, 60, 64] do not effectively leverage all available reasoning signals in these trajectories. In effect, these methods treat each *reasoning* trajectory independently, aggregate final *answers* for a vote, and leave unexploited the potentially complementary reasoning encoded across the

*Equal contribution. Correspondence to zeming.chen@epfl.ch

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

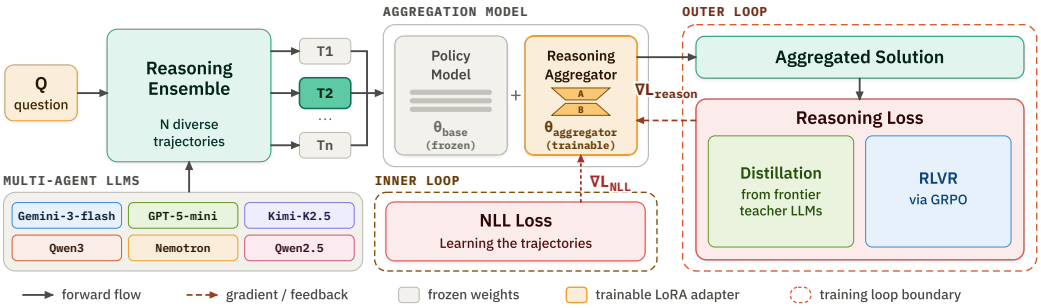


Figure 1: **Meta-learning `TTT-Consolidate` for reasoning consolidation.** The training procedure involves a nested inner and outer loop. The inner loop optimizes the likelihood of a batch of reasoning trajectories with respect to the parameters of the LoRA reasoning aggregator. In the outer loop, the model uses the merged reasoning information in the aggregator to synthesize the final solution. In both loops, only the LoRA is updated while the base LLM is frozen.

full ensemble of trajectories [22]. Although recent work proposes model-based in-context aggregation to better harness the reasoning across trajectories [10, 45, 73], these are attention-based aggregates. Studies of long-context language models show that access to information does not guarantee its effective use, and these methods still face typical long-context degradation issues [17, 27, 29].

We propose `TTT-Consolidate`, a meta-learned framework that transforms trajectory aggregation from passive selection into active consolidation as test-time learning. At inference, `TTT-Consolidate` consolidates diverse reasoning trajectories in a lightweight LoRA adapter via gradient updates and then synthesizes a final solution from the adapted model. We train the adapter via bi-level optimization; the inner loop minimizes a self-supervised next-token-prediction loss over a batch of trajectories, while the outer loop is trained to produce a synthesized answer, either through supervised finetuning on demonstrations distilled from a frontier teacher or through RLVR with GRPO [16, 47]. The outer-loop objective backpropagates through the inner loop to shape how the LoRA aggregates the ensemble end-to-end, teaching the adapter to consolidate trajectories into representations that improve reasoning.

We train `TTT-Consolidate` on *OpenParallelThinking*, a novel dataset containing 13K math problems, each paired with groups of reasoning trajectories from 6 LLMs. Across math and STEM benchmarks, `TTT-Consolidate` outperforms all rule- and model-based aggregation methods, with average gains of up to 20% over the strongest baselines. These gains widen as model scale increases, and remain strong even under matched compute and runtime settings, which normally disfavor test-time training methods. Importantly, `TTT-Consolidate` learns from the reasoning behavior in the trajectories, rather than shortcutting to voting based on the final answers in trajectories. In particular, on weak mixtures where no input trajectory is correct, `TTT-Consolidate` can recover from these unsuccessful trajectories and arrive at an improved solution, outperforming the best baseline by up to 23%, and its gains scale with the number of incorrect trajectories in consolidation. Finally, RL further improves the adapted policy's ability to reason over consolidated trajectories, adding an average 5-point gain over `TTT-Consolidate` trained with SFT.

2 Background and Preliminaries

2.1 Test-Time Scaling and Reasoning Aggregation

Scaling inference-time compute complements scaling parameters or pre-training data for improving LLM reasoning [64]. Among test-time scaling strategies, *parallel thinking*, i.e., sampling multiple reasoning trajectories and aggregating them into a single answer, has yielded remarkable empirical gains [24]. Formally, given a question Q with verifiable ground-truth answer y^*, parallel thinking involves sampling N reasoning trajectories $\mathcal{R} = \{R_1, \ldots, R_N\}$ from a causal language model $p_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$, where each R_n is a complete chain of thought ending in a candidate answer a_n. An *aggregator* maps the question and trajectories to a final prediction. We write this as a generic operator: $\hat{y} = \mathcal{A}(Q, \mathcal{R}; \boldsymbol{\theta})$. The design space of $\mathcal{A}$ has evolved through three progressively more expressive paradigms.

Rule-Based Aggregation. Standrad aggregations apply hand-crafted rules to conclude a final answer from R. *Majority Voting* [60] returns the modal answer, while *reward-guided selection* [64] ranks trajectories with an external scorer r and keeps the top one:

$$\hat{y}_{\text{rule}} = \begin{cases} \text{mode}\big(\{a_n\}_{n=1}^{N}\big) & \text{(majority vote)} \\ a_{n^\star}, \;\; n^\star = \arg\max_{n\in[N]} r(Q,\, R_n) & \text{(reward selection)} \end{cases} \tag{1}$$

Rule-based aggregation discards intermediate reasoning, so partial correctness in trajectories with wrong final answers (e.g., a correct sub-derivation or a pruned but valid case) is never leveraged.

Model-Based In-Context Aggregation. To exploit intermediate reasoning, recent work concatenates the trajectories into the prompt and produces an aggregated answer through a single forward pass with *frozen* parameters [54, 56]: $\hat{y}_{\text{ctx}} \sim p_{\boldsymbol{\theta}}\big(\cdot \mid Q{\cdot}R_1{\cdot}R_2{\cdot}\cdots{\cdot}R_N\big)$. Information from $\mathcal{R}$ enters the model only through self-attention over the context. As N grows, the context length scales linearly, and LLMs are well documented to degrade on long, distractor-rich inputs [17, 27, 29, 31, 66], leaving the rich complementary reasoning in $\mathcal{R}$ underexploited.

RL-Trained Model Aggregation. A natural refinement keeps the in-context interface but *trains* the aggregator end-to-end with reinforcement learning from verifiable rewards (RLVR), maximizing the expected reward of the aggregated output [54, 73]:

$$\boldsymbol{\theta}^{\text{RL}} = \arg\max_{\boldsymbol{\theta}} \; \mathbb{E}_{(Q,\,\mathcal{R},\,y^*)}\Big[\mathbb{E}_{\hat{y}\sim p_{\boldsymbol{\theta}}(\,\cdot\,\mid Q,\,\mathcal{R})}\big[r(\hat{y},\, y^*)\big]\Big]. \tag{2}$$

This turns aggregation into a learned skill rather than a fixed heuristic, but still consumes $\mathcal{R}$ only through attention over a concatenated context, where the same long-context bottleneck persists.

2.2 Test-Time Training

A common thread in prior paradigms is that $\boldsymbol{\theta}$ is fixed at inference, and trajectories are processed only through context. In this work, we propose to perform aggregation with *Test-time training* (TTT), consolidating the trajectories $\mathcal{R}$ directly in the model's parameters via S gradient steps using a self-supervised language modeling objective $\mathcal{L}_{\text{NLL}}$:

$$\phi^{(s)} = \phi^{(s-1)} - \alpha\, \nabla_{\phi^{(s-1)}} \mathcal{L}_{\text{NLL}}\big(\phi^{(s-1)};\, \mathcal{R}\big), \qquad s = 1, \ldots, S, \tag{3}$$

starting from $\phi^{(0)} = \boldsymbol{\theta}$ and yielding adapted parameters $\phi(\boldsymbol{\theta}, \mathcal{R}) \triangleq \phi^{(S)}$. The aggregated final prediction is then drawn from the *updated* model: $\hat{y}_{\text{TTT}} \sim p_{\,\phi(\boldsymbol{\theta},\,\mathcal{R})}\big(\cdot \mid Q\big)$. This formulation is motivated by the observation that LLMs store and retrieve knowledge more reliably through their parameters than long-range attention [13, 50, 76], sidestepping the long-context bottleneck that limits model-based aggregation.

Meta-Learning for Test-Time Training. Training an LLM for TTT is naturally a meta-learning problem [12, 13, 20, 55], where initial parameters $\boldsymbol{\theta}$ are meta-learned such that gradient-based adaptation of any $\mathcal{R}$ at test time yields strong downstream performance. With $\phi(\boldsymbol{\theta}, \mathcal{R})$ denoting the post-adaptation parameters, the meta-objective is the bilevel optimization

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \; \mathbb{E}_{(Q,\,\mathcal{R},\,y^*)}\Big[\mathcal{L}_{\text{inner}}\big(\phi(\boldsymbol{\theta}, \mathcal{R});\; Q,\; y^*\big)\Big], \tag{4}$$

where $\mathcal{L}_{\text{inner}}$ governs how $\mathcal{R}$ is internalized and $\mathcal{L}_{\text{outer}}$ scores the adapted model on the downstream task. Following common practice [13, 32, 50], $\mathcal{L}_{\text{inner}}$ is the negative log-likelihood (NLL) over $\mathcal{R}$,

$$\mathcal{L}_{\text{inner}}(\phi;\, \mathcal{R}) = -\sum_{k=1}^{N} \sum_{t=1}^{|R_n|} \log p_\phi\big(R_{n,t} \mid R_{n,<t}\big), \tag{5}$$

which encodes the reasoning traces into the model's parameters. The outer-loop loss admits two natural instantiations:

$$\mathcal{L}_{\text{outer}}(\phi;\, Q,\, y^*) = \begin{cases} -\sum_{t=1}^{T_{y^*}} \log p_\phi\big(y^*_t \mid Q,\, y^*_{<t}\big) & \text{(SFT)} \\ -\,\mathbb{E}_{\,y\sim p_\phi(\,\cdot\,\mid Q)}\big[r(y,\, y^*)\big] & \text{(RLVR)} \end{cases} \tag{6}$$

The SFT objective provides a dense, low-variance signal but requires annotated demonstrations, while the RLVR objective directly optimizes the verifiable performance at the cost of higher variance. `TTT-Consolidate` builds on this formulation by instantiating the inner loop as a parameter-efficient (LoRA) adaptation that consolidates $\mathcal{R}$, and the outer loop as a learned synthesis of a final chain-of-thought (CoT) solution from the adapted model.

3 `TTT-Consolidate`: Reasoning Consolidation as Test-Time Learning

We instantiate the **Test-Time Training** (TTT) framework from Section 2 for reasoning aggregation. Rather than treating $\mathcal{R}$ as a passive context, `TTT-Consolidate` consolidates the trajectories into LoRA adapter weights ϕ [28] attached to a frozen backbone $\boldsymbol{\theta}_{\text{base}}$ via gradient-based inner-loop adaptation, then answers Q using the adapted model. Figure 1 shows the procedure.

Inner Loop Consolidation. For each training example $(Q, \mathcal{R}, y^*)$, the inner loop initializes $\phi^{(0)} = \boldsymbol{\theta}_{\text{adapter}}$ and adapts only ϕ while keeping $\boldsymbol{\theta}_{\text{base}}$ fixed:

$$\phi^{(s)} = \phi^{(s-1)} - \alpha \nabla_{\phi} \mathcal{L}_{\text{inner}}\left(\phi^{(s-1)}; Q, \mathcal{R}\right), \qquad s = 1, \ldots, S, \tag{7}$$

where the consolidation loss is the question-conditioned NLL of the sampled reasoning trajectories,

$$\mathcal{L}_{\text{inner}}(\phi; Q, \mathcal{R}) = -\sum_{k=1}^{K} \sum_{t=1}^{|R_k|} \log p_{\boldsymbol{\theta}_{\text{base}} \oplus \phi}(R_{k,t} \mid Q, R_{k,<t}). \tag{8}$$

Thus, the adapted $\phi^{(S)}$ serves as a parametric consolidation of $\mathcal{R}$. At inference time, the final answer is generated from Q alone using $p_{\boldsymbol{\theta}_{\text{base}} \oplus \phi^{(S)}}(\cdot \mid Q)$.

Outer Loop Meta-Learning The outer loop optimizes $\boldsymbol{\theta}_{\text{adapter}}$ by differentiating the meta objective through the inner-loop updates. For SFT, we use the first branch of Eq. (6) with p_{ϕ} replaced by $p_{\boldsymbol{\theta}_{\text{base}} \oplus \phi^{(S)}}$. The adapted model is trained using solution demonstrations from frontier teacher models, GPT-5.4-`xhigh` and Gemini 3.1 Pro Deep Think, which provide high-quality CoT solutions as supervised targets [40, 51, 52].

For RLVR, we use the GRPO loss over the adapted policy (base model with adapted LoRA):

$$\begin{aligned} \mathcal{L}_{\text{GRPO-MAML}} &= -\frac{1}{G} \sum_{i=1}^{G} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min\left(r_{i,t} \hat{A}_i,\ \text{clip}(r_{i,t}, 1-\epsilon, 1+\epsilon) \hat{A}_i\right), \\ r_{i,t} &= \frac{p_{\boldsymbol{\theta}_{\text{base}} \oplus \phi^{(S)}}(y_{i,t} \mid Q, y_{i,<t})}{p_{\boldsymbol{\theta}_{\text{base}} \oplus \phi^{(S)}_{\text{old}}}(y_{i,t} \mid Q, y_{i,<t})}. \end{aligned} \tag{9}$$

Here $\{y_i\}_{i=1}^{G}$ are sampled responses from the adapted policy, $\hat{A}_i$ are group-normalized advantages, and $\phi^{(S)}_{\text{old}}$ is the frozen reference copy of the adapted LoRA weights. To reduce memory, we apply truncated gradient unrolling (Appendix D).

3.1 The *OpenParallelThinking* Dataset

Existing reasoning datasets typically pair each problem with one solution, leaving little supervision for consolidating multiple reasoning traces. We address this gap with *OpenParallelThinking*, a dataset for training reasoning aggregators.

OpenParallelThinking targets three design goals: source diversity, trajectory diversity, and balanced difficulty. We pool problems from four established math corpora and deduplicate them, then generate trajectories from four solvers spanning different scales, families, and training recipes, so the aggregator is not biased toward a single reasoning style. Trajectories are grouped per question under a token budget, and groups are filtered by correctness ratio to emphasize aggregation over predominantly incorrect trajectories, which yields a stronger learning signal than relying on a majority-correct shortcut (Figure 2). Appendix C.2 gives the full construction pipeline, including source datasets, solver selection, deduplication, grouping, and filtering. The final release contains 12,810 questions and 63,507 trajectory groups, drawn from 19,615 unique deduplicated problems. We sample 8,000 instances for RL training and reserve the rest for SFT. We will release the dataset and construction pipeline to support future work on reasoning aggregation.

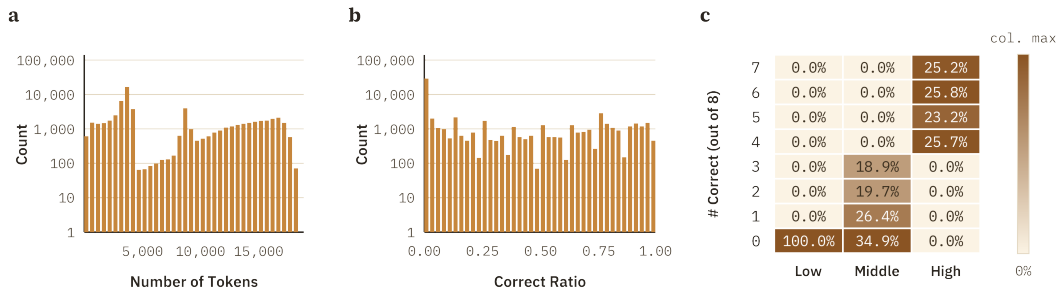


Figure 2: **Left:** Token length distribution of the *OpenParallelThinking*. **Center:** Correct ratio distribution *OpenParallelThinking*, reflecting a cosine-like difficulty profile shaped by correctness-based filtering. **Right:** Correctness distribution of the evaluation trajectory groups across three tiers.

4 Experiments

4.1 Experimental Setup

Benchmarks for Reasoning Aggregation. We evaluate on reasoning-heavy, meaningful benchmarks across two domains. For mathematical reasoning, we use AIME2025 [2], AIME2026 [3], Apex2025, ArxivMath, CMIMC2025, HMMT-feb-2026, and HMMT-nov-2025 from MathArena [4], and IMO-Bench [39]. For generalization beyond math, we use two general STEM benchmarks: GPQA-diamond [46] and TheoremQA [9]. For every problem, we generate eight reasoning trajectories (each ends with a final answer) distilled from LLMs spanning a range of capability levels. The generation and filtering are detailed in Appendix C.1. We group these trajectories into a quality bucket that isolates the effect of trajectory quality on aggregation performance: *Minority* (0–3 correct). Essentially, having ≥ 4 correct trajectories already allows a simple rule-based aggregation like Majority Voting to get 100% accuracy, which is unlikely to occur naturally. We additionally group trajectories into *None*, where all trajectories are incorrect, to analyze whether `TTT-Consolidate` can learn from their partial reasoning and aggregate them into a more reliable solution.

Baselines. We compare our method with four reasoning aggregation methods. For rule-based aggregation, we evaluate majority voting [60]. For model-based aggregation, we compare with three inference-time in-context aggregation variants: prompt aggregation, SFT aggregation, and RL aggregation [73]. Prompt aggregation uses the base model directly in a zero-shot manner, whereas SFT and RL aggregation use models trained with the corresponding objectives. We use Qwen3-1.7B and Qwen3-4B [67] as the base models, and for all small datasets (less than 100 samples), and run them 8 times individually and report their average scores. For non-aggregation methods, we compare with the standard reasoning-heavy R1-style inference [25] with a 16K token extended thinking budget. See Appendix D, E for other implementation details.

4.2 Main Results

`TTT-Consolidate` consistently outperforms baseline aggregation methods. Across ten benchmarks and both trajectory-quality tiers, `TTT-Consolidate` outperforms the baseline aggregation methods in nearly every case (Table 1). At fixed model scale, the gain is stronger in the more difficult *None* tier, where `TTT-Consolidate`'s advantage grows precisely where input trajectories all contain incorrect conclusions and rule-based aggregation fails. At a fixed tier, scaling from 1.7B to 4B roughly doubles the gain, indicating that a stronger aggregator extracts more signal from the same set of trajectories. Compared to extended thinking with a 16K-token budget, `TTT-Consolidate` trails by only 4.9 (1.7B) and 4.5 (4B) points when only aggregating incorrect trajectories (*None*). With *Minority*, `TTT-Consolidate` surpasses extended thinking by 7 (1.7B) and 10 (4B) points, with the gap widening at a larger scale. In contrast, every baseline aggregator trails extended thinking by a large margin, even in the *Minority* setting. Notably, SFT Aggregation is trained on the same questions, trajectory groups, and frontier-generated target solutions as `TTT-Consolidate`, so this gap cannot be explained by frontier supervision in the outer loop alone: `TTT-Consolidate`'s consolidation mechanism is what drives the improvement.

Table 1: **Main results (%) on math and STEM reasoning benchmarks.** Each row group corresponds to a model scale and trajectory quality setting. *None*: All 8 trajectories are incorrect. *Minority*: A minority of trajectories are correct. **Bold** shows the best result per column within each group. Underline reflects the best performance across all groups.

		Math Reasoning								STEM	
Setting	**Method**	**AIME25**	**AIME26**	**Apex25**	**ArXivMath**	**CMIMC25**	**HMMT-2602**	**HMMT-2511**	**IMO-Bench**	**GPQA**	**TheoremQA**
Extended Thinking	Qwen3-1.7B	40.0	40.8	0.0	13.9	25.3	21.9	25.8	22.8	37.4	57.5
	Qwen3-4B	71.7	68.3	0.0	16.7	52.2	40.5	60.4	37.0	51.5	68.6
Rule-based Agg.	Majority Voting @8 **(None)**	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	Majority Voting @8 **(Minority)**	26.7	23.3	25.0	18.1	30.0	30.3	16.7	21.5	16.2	19.5
Qwen3-1.7B None	Prompt Aggregation (CoT)	5.0	6.7	0.0	0.0	0.0	2.7	2.1	1.5	7.1	14.6
	Prompt Aggregation (Thinking)	22.9	21.7	0.0	2.8	10.9	17.8	7.9	7.5	13.6	25.1
	SFT Aggregation (CoT)	2.5	5.4	0.0	1.4	2.5	4.2	3.3	0.8	0.5	11.1
	RL Aggregation (Thinking)	2.9	6.7	0.0	2.8	0.3	0.4	2.9	2.0	7.1	13.3
	TTT-Consolidate†	**36.6**	**26.7**	**16.7**	**23.6**	**20.0**	**18.2**	**23.3**	**14.5**	**22.7**	**33.5**
Qwen3-1.7B Minority	Prompt Aggregation (CoT)	23.3	15.8	0.0	13.9	13.8	15.2	20.0	15.0	28.3	26.4
	Prompt Aggregation (Thinking)	36.3	**42.1**	3.1	16.7	28.1	25.8	21.7	18.5	28.8	37.3
	SFT Aggregation (CoT)	13.3	14.6	0.0	4.2	12.5	12.1	4.6	7.8	15.2	18.1
	RL Aggregation (Thinking)	31.3	25.4	0.0	19.5	23.8	22.4	23.8	17.3	28.8	27.3
	Recursive Refinement	26.7	18.9	0.0	11.1	21.7	18.2	14.4	23.5	26.8	–
	TTT-Consolidate†	**43.3**	30.0	**25.0**	**34.7**	**45.0**	**27.3**	**36.4**	**25.5**	**42.9**	**43.6**
Qwen3-4B None	Prompt Aggregation (CoT)	11.7	4.6	0.0	4.2	4.4	9.1	6.7	3.0	10.1	21.3
	Prompt Aggregation (Thinking)	32.9	36.7	0.0	5.6	24.5	24.6	14.2	12.0	17.2	28.6
	SFT Aggregation (CoT)	16.7	17.9	0.0	4.2	3.4	4.9	6.7	3.3	2.0	17.6
	RL Aggregation (Thinking)	15.8	15.8	16.7	5.6	15.0	9.1	20.0	6.7	4.3	32.6
	TTT-Consolidate†	**69.5**	**39.1**	**33.3**	**37.9**	**35.0**	**48.5**	**52.3**	**28.3**	**22.5**	**41.9**
Qwen3-4B Minority	Prompt Aggregation (CoT)	28.3	26.7	0.0	16.7	31.9	25.0	26.7	18.0	29.3	33.4
	Prompt Aggregation (Thinking)	61.7	**61.7**	2.1	23.6	44.1	26.1	35.8	23.8	30.8	39.8
	SFT Aggregation (CoT)	30.0	23.3	8.3	6.9	15.0	18.2	20.0	7.0	3.5	22.0
	RL Aggregation (Thinking)	56.7	53.3	8.3	23.6	40.0	28.0	38.4	22.3	21.7	37.6
	TTT-Consolidate†	**69.6**	47.6	**41.6**	**42.9**	**78.6**	**57.7**	**72.7**	**38.3**	**64.6**	**61.9**

`TTT-Consolidate` can aggregate unsuccessful trajectories into a more reliable solution. In the *None* setting, every input trajectory arrives at an incorrect final solution, forcing majority voting to 0% performance. Averaged across all ten benchmarks, `TTT-Consolidate` achieves absolute gains of 22.6 and 10.6 points, approximately doubling the strongest baseline performance (Figure 3). The results suggest that test-time learning to consolidate can encode useful partial reasoning from collectively wrong attempts, producing answers stronger than individual attempts or the aggregation model solving the problem alone.[2]

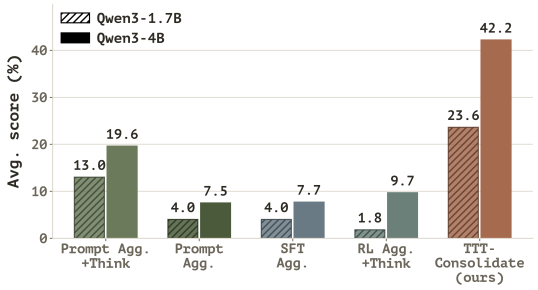


Figure 3: **Aggregating collectively incorrect trajectories (*None*) with `TTT-Consolidate`.** With both 1.7B and 4B models, `TTT-Consolidate` improves the most compared to other methods.

`TTT-Consolidate` demonstrates out-of-domain transfer. `TTT-Consolidate` is trained only on math reasoning from *OpenParallelThinking* we constructed, but its advantage generalizes cleanly to scientific domains. Across GPQA and TheoremQA, `TTT-Consolidate` averages 14 points above the strongest baseline across trajectory quality tiers and model scales, comparable in magnitude to its in-domain mathematical gains, suggesting consolidation based on test-time learning is a domain-general reasoning operation.

[2]Qualitatively, we observe that `TTT-Consolidate` can consolidate failed attempts into an improved solution. We use three frontier LLM judges (Fable 5, Opus 5, GPT-5.6-Sol; extra-high) to compare 100 `TTT-Consolidate` solutions with their *None* tier input trajectories. 98% of solutions show clear synthesis of a better reasoning trajectory from the incorrect inputs. An audit of a separate 100 *Minority*-tier solutions finds only 1 case of copying the input trajectory. Appendix C.3 shows the judge protocol and details.

Table 3: **Accuracy vs. compute** on the *None* tier, averaged across the three benchmarks. K denotes the number of input trajectories, all of which have incorrect final answers. We report accuracy (%) and total FLOPs for trajectory sampling and aggregation. The in-context FLOPs apply to all in-context aggregation methods (prompt-only, SFT, RL). `TTT-Consolidate` achieves higher accuracy even when the baselines use twice as many trajectories and more compute.

K	MV		In-context Aggregation Methods				`TTT-Consolidate`	
	FLOPs	Acc.	FLOPs	Prompt Acc.	SFT Acc.	RL Acc.	FLOPs	Acc.
2	1.755e14	0.0	2.227e14	4.3	2.2	1.9	2.897e14	**15.2**
4	3.509e14	0.0	4.105e14	3.4	3.9	1.7	5.432e14	**23.0**
8	7.019e14	0.0	7.890e14	5.5	3.7	2.4	1.050e15	**23.6**
16	1.404e15	0.0	1.558e15	4.7	6.7	0.8	2.064e15	**26.5**

5 Analysis

5.1 `TTT-Consolidate`'s performance scales as the number of weak trajectories increases

We analyze how aggregation methods scale with the number of weak trajectories under the *None* tier. With all trajectories being incorrect, we rule out the shortcut of selecting an already-correct candidate, directly testing whether each method can extract useful knowledge from wrong attempts. We grow the number of trajectories as $n \in \{2, 4, 8, 16\}$, with Qwen3-1.7B as the backbone.

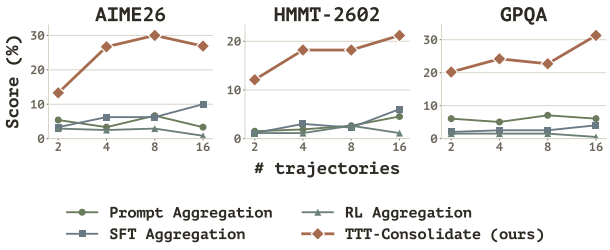


Figure 4: **Scaling the number of weak trajectories under aggregation.** As more *None*-tier weak trajectories are aggregated, baseline aggregations improve slightly or degrade while `TTT-Consolidate` continues to improve.

We show in Figure 4 that as more incorrect trajectories are added, the baseline performance stagnates or even degrades. In contrast, `TTT-Consolidate` generally benefits from more trajectories during consolidation, with, for instance, an 11.1% increase on `GPQA`. We hypothesize that this stems from test-time learning deriving informative signal from noisy additional trajectories.

5.2 `TTT-Consolidate` outperforms compute-matched baselines

`TTT-Consolidate` outperforms majority voting at matched compute. We verify if `TTT-Consolidate`'s gain comes from the extra test-time compute by checking if a compute-matched baseline could achieve a similar performance. We allocate `TTT-Consolidate`'s TTL budget to additional majority-voted rollouts (Table 2). Even with 1.50× more FLOPs, majority voting @18 is still worse by 12.8 points than `TTT-Consolidate`@8, and does not meaningfully improve as the compute grows (22.7 → 22.2 → 22.6). Appendix A extends this comparison to reward-guided and recursive-refinement baselines on latency and memory.

Table 2: **Majority voting vs. `TTT-Consolidate` on matched compute budget**. MV matches `TTT-Consolidate`'s compute budget by spending on extra rollouts. MV does not improve with more rollouts and stays well below `TTT-Consolidate`, even with more compute.

Method	FLOPs	Relative Compute	Minority Acc.
MV @8	7.019e14	0.67×	22.7
MV @13	1.141e15	1.09×	22.2
MV @18	1.579e15	1.50×	22.6
`TTT-Consolidate` @8	**1.050e15**	**1.00×**	**35.4**

`TTT-Consolidate` outperforms in-context aggregation even when using less compute. We further evaluate accuracy as a function of total computation, sweeping the number of incorrect trajectories K (*None*). We report average accuracy across the three benchmarks in Figure 4 with the FLOPs for trajectory sampling and reasoning aggregation. `TTT-Consolidate` consistently outperforms every aggregation baseline even when they use 1.5× more FLOPs and twice as many trajectories. `TTT-Consolidate`'s performance increases with compute, from 15.2 to 26.5 when

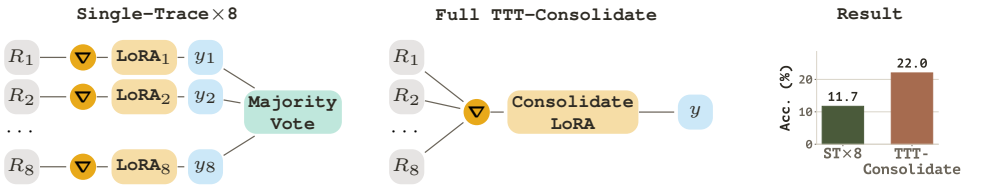


Figure 5: **Does trajectory information interact in the consolidation step?** Under Single-Trace×8 (left), each trajectory is adapted and decoded independently, then reconciled by a vote. Under full `TTT-Consolidate` (middle), all 8 trajectories jointly update one LoRA before any answer is generated. Single-Trace×8 underperforms full `TTT-Consolidate` on the *None* tier (right).

K grows to 16. Majority voting always scores 0.0 since all candidates are wrong. Equal or greater FLOPs on additional sampling or in-context aggregation do not match `TTT-Consolidate`'s performance. `TTT-Consolidate`'s gains arise from consolidating useful intermediate reasoning through test-time learning, not simply from more test-time computation.

5.3 `TTT-Consolidate` requires joint consolidation in parameter space

We validate that `TTT-Consolidate`'s gains require cross-trajectory interaction, not repeated exposure to the same question or task-distribution activation. We compare `TTT-Consolidate` to using 8 separately adapted LoRAs, one per input trajectory, with a majority vote over their 8 generated solutions (**Single-Trace×8**; Figure 5). Both see the same 8 trajectories and spend matched adaptation compute. The only difference is whether cross-trajectory interaction happens before producing a solution. Single-Trace×8 is worse by 10.3 points at matched compute, showing that `TTT-Consolidate`'s gain requires consolidating trajectories *jointly* within a shared parameter space (adapted LoRA) rather than encoding them independently and reconciling only by voting over outputs. Appendix B.2 extends this comparison with two more controls, passive question repetition and task-distribution activation without target-specific trajectories, both of which perform worse than Single-Trace×8, ruling them out as alternative explanations.

5.4 `TTT-Consolidate` consolidates reasoning rather than voting on the answers

An aggregation method that scores well when most trajectories carry the correct answer could either be consolidating reasoning or simply voting on final answers. To distinguish these, we use the *Majority* quality bucket: 4 to 7 trajectories carry correct answers. We swap the correct/incorrect answers so the majority answer becomes wrong, while reasoning stays untouched (*Majority-reversed*). A consolidator should be largely invariant, but a "voter" should collapse. We show the results in Figure 6.

Under the *Majority* tier, Prompt and RL Aggregation reach 80–90% on several benchmarks. However, after reversal, RL drops by up to 64 points and Prompt by up to 52 points, often falling below 30%. The large drop suggests their high scores came from shortcut voting on final answers rather than understanding the trajectories. In contrast, `TTT-Consolidate`'s drop stays within the $[-17, -5]$ range across all benchmarks, indicating it utilizes information from trajectories rather than counting answers.

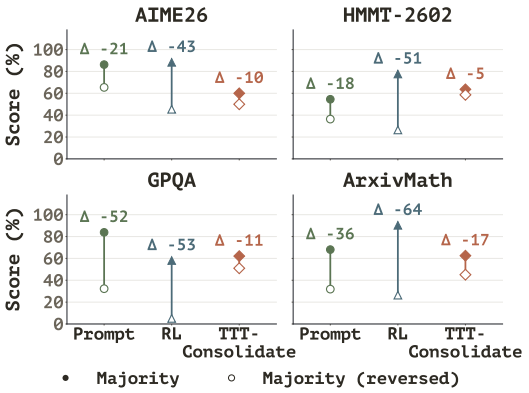


Figure 6: **Robustness to majority manipulation.** Dumbbells connect each method's score under *Majority* (filled) and *Majority-reversed* (open, correct/incorrect answers swapped); bar length is the resulting drop. Baselines collapse while our method stays short in every panel, indicating consolidation rather than answer-tallying.

5.5 RL incentivizes `TTT-Consolidate`'s reasoning over consolidated trajectories

Reinforcement learning with verifiable rewards (RLVR) has become a standard recipe for eliciting stronger reasoning from LLMs [25]. A natural question is whether such reward-driven optimization can also improve *reasoning consolidation*. We cast RLVR as the outer-loop task for meta-learning over `TTT-Consolidate` (Qwen3-1.7B) where the inner loop performs consolidation, and the outer loop uses GRPO to maximize the likelihood that the consolidated policy reaches a correct answer [47].

Figure 7 compares `TTT-Consolidate` (RL-enhanced) against the supervised `TTT-Consolidate` on aggregating *None* tier trajectories. RL training lifts the average accuracy by 4.6 absolute gain (a $\sim 22\%$ relative improvement), with the largest gains on GPQA-Diamond, nearly doubling from 22.7 to 40.0 (+17.3). RL improves the final adapted policy's ability to reason over consolidated trajectories, while its current training gains arise mainly from a stronger initialization for consolidation.

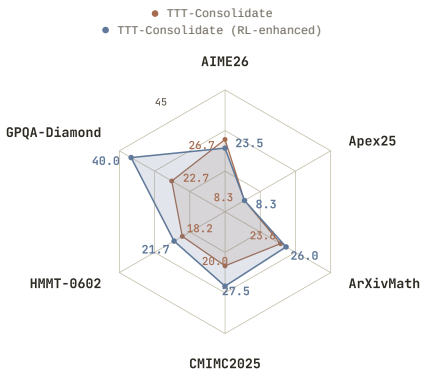


Figure 7: **`TTT-Consolidate` enhanced with RLVR** on *None* tier trajectories, shows consistent gains over `TTT-Consolidate` trained with SFT.

6 Related Work

Reasoning Aggregation. A common way to combine multiple LLM solutions is rule-based voting: self-consistent decoding [7, 59] samples multiple reasoning trajectories and returns the most frequent answer, with later variants adapting the procedure through dynamic sampling or heuristic filtering [1, 8, 57]. This strategy breaks down when correct solutions occupy minority modes, causing voting to amplify errors rather than correct them. Learned models offer an alternative: reward models score and rank candidates [65, 70], Universal Self-Consistency [10] has the model inspect all sampled answers and select the most consistent one, and Zhao et al. [73] instead train the model to reason over and synthesize across candidates as an explicit skill. However, these methods all concatenate candidates directly into the context, suffering from attention dilution and rising cost as context grows. In contrast, `TTT-Consolidate` meta-learns to proactively internalize cross-trajectory information rather than read over the candidate solutions.

Parallel Thinking. Parallel thinking has emerged as a promising paradigm for overcoming the fragility of single-trajectory sequential reasoning [26, 35, 61, 63, 69]. Most recent work trains models to natively generate and manage multiple thought branches concurrently: Parathinker [62] generates diverse, independent thoughts end-to-end and aggregates them into a final answer, while Parallel-R1 [74] first instills this ability via SFT on easy math problems (cold-start) before extending it to harder problems through reinforcement learning. Other approaches instead refine the structure of parallel reasoning branches [41, 68]; notably, Venkatraman et al. [54] frame the process as an evolutionary loop that recursively aggregates and refines populations of candidate solutions across iterations. Our method differs by decoupling generating parallel thoughts from reasoning over them, enabling the aggregation of thoughts from different sources beyond self-generated ones.

Test-Time Training. MAML [21] introduces optimization-based, model-agnostic meta-learning for efficient downstream adaptation; Chen et al. [12] apply MAML to GPT models, internalizing context knowledge via inner-loop gradient updates before answering queries in the outer loop. A related line of work instead adds or repurposes weight modules that mimic global sequence modeling while avoiding sequential dependency and quadratic scaling [15, 19, 71], such as Sun et al.'s [49] RNN layers with MLP-based hidden states, and new architectures combining persistent neural memory with standard attention to extend context length while preserving context-grounding [5, 6]. Our method, `TTT-Consolidate`, defines reasoning aggregation as a test-time learning process.

7 Conclusion

We present `TTT-Consolidate`, a meta-learned reasoning aggregation that actively consolidates thinking trajectories using test-time learning. Rather than reading trajectories as context,

TTT-Consolidate consolidates them in a lightweight LoRA adapter through a learned adaptation process, enabling the adapted LoRA to synthesize a more reliable solution. To enable training and evaluation at scale, we introduce *OpenParallelThinking*, a corpus of 13K problems paired with 63K trajectory groups spanning diverse models and three quality levels. Across ten challenging math and STEM benchmarks, TTT-Consolidate outperforms the best aggregation baselines by 20% on average, with particularly notable gains when all input trajectories are incorrect, confirming that TTT-Consolidate can recover from unsuccessful trajectories and arrive at an improved solution through aggregation as test-time learning. Further analysis shows that TTT-Consolidate resists majority-voting shortcuts, scales positively with weak trajectory count, and that RL further improves the adapted policy's ability to reason over consolidated trajectories.

Acknowledgments

We gratefully acknowledge the support of the Swiss National Science Foundation (No. 215390), the AI2050 program at Schmidt Sciences (Grant #G-25-69783), Sony Group Corporation, and the Swiss National Supercomputing Center (CSCS) in the form of an infrastructure engineering and development project. This research is with support from Google.org and the Google Cloud Research Credits program for the Gemini Academic Program. This project is also funded by the European Union (ERC, RESPECT-LM, 101222478). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.761. URL https://aclanthology.org/2023.emnlp-main.761/.

[2] AI-MO. Ai-mo/aimo-validation-aime. https://huggingface.co/datasets/AI-MO/aimo-validation-aime, 2025. 2025.

[3] AI-MO. Ai-mo/aimo-validation-aime. https://huggingface.co/datasets/AI-MO/aimo-validation-aime, 2026. 2026.

[4] Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions, February 2025. URL https://matharena.ai/.

[5] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

[6] Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025.

[7] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[8] Weizhe Chen, Sven Koenig, and Bistra Dilkina. Lspo: Length-aware dynamic sampling for policy optimization in llm reasoning. *arXiv preprint arXiv:2510.01459*, 2025.

[9] Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. Theoremqa: A theorem-driven question answering dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7889–7901, 2023.

[10] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation. *arXiv preprint arXiv:2311.17311*, 2023.

[11] Yang Chen, Zhuolin Yang, Zihan Liu, Chankyu Lee, Peng Xu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acereason-nemotron: Advancing math and code reasoning through reinforcement learning. *arXiv preprint arXiv:2505.16400*, 2025.

[12] Zeming Chen, Gail Weiss, Eric Mitchell, Asli Celikyilmaz, and Antoine Bosselut. Reckoning: reasoning through dynamic knowledge encoding. *Advances in Neural Information Processing Systems*, 36:62579–62600, 2023.

[13] Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. *arXiv preprint arXiv:2507.06415*, 2025. URL https://arxiv.org/abs/2507.06415.

[14] 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 arXiv:2110.14168*, 2021.

[15] Karan Dalal, Daniel Koceja, Jiarui Xu, Yue Zhao, Shihao Han, Ka Chun Cheung, Jan Kautz, Yejin Choi, Yu Sun, and Xiaolong Wang. One-minute video generation with test-time training. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17702–17711, 2025.

[16] DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[17] Shihan Dou, Ming Zhang, Zhangyue Yin, Chenhao Huang, Yujiong Shen, Junzhe Wang, Jiayi Chen, Yuchen Ni, Junjie Ye, Cheng Zhang, Huaibing Xie, Jianglu Hu, Shaolei Wang, Weichao Wang, Yanling Xiao, Yiting Liu, Zenan Xu, Zhen Guo, Pluto Zhou, Tao Gui, Zuxuan Wu, Xipeng Qiu, Qi Zhang, Xuanjing Huang, Yu-Gang Jiang, Di Wang, and Shunyu Yao. CL-bench: A benchmark for context learning. *arXiv preprint arXiv:2602.03587*, 2026. URL https://arxiv.org/abs/2602.03587.

[18] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2023. URL https://arxiv.org/abs/2305.14325.

[19] Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Di He, Wenhao Huang, and Tianle Cai. In-place test-time training. *arXiv preprint arXiv:2604.06169*, 2026.

[20] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[21] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[22] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv preprint arXiv:2508.15260*, 2025.

[23] Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. *arXiv preprint arXiv:2410.07985*, 2024.

[24] Google DeepMind. Advanced version of Gemini with Deep Think officially achieves gold-medal standard at the international mathematical olympiad. https://deepmind.google/blog, July 2025. Accessed: 2026.

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

[26] Qi Guo, Jianing Wang, Deyang Kong, Xiangyu Xi, Jianfei Zhang, Yi Lu, Jingang Wang, Wei Wang, Shikun Zhang, and Wei Ye. Ope: Overcoming information saturation in parallel thinking via outline-guided path exploration. *arXiv preprint arXiv:2602.08344*, 2026.

[27] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

[28] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[29] Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554, 2024.

[30] 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.

[31] Mosh Levy, Alon Jacoby, and Yoav Goldberg. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15339–15353, 2024.

[32] Gaotang Li, Ruizhong Qiu, Xiusi Chen, Heng Ji, and Hanghang Tong. Beyond log likelihood: Probability-based objectives for supervised fine-tuning across the model capability continuum. *arXiv preprint arXiv:2510.00526*, 2025.

[33] Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. Numinamath. [https://huggingface.co/AI-MO/NuminaMath-CoT](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024.

[34] 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*, 2023.

[35] Houjun Liu, Shikhar Murty, Christopher D Manning, and Róbert Csordás. Thoughtbubbles: an unsupervised method for parallel thinking in latent space. *arXiv preprint arXiv:2510.00219*, 2025.

[36] Zihan Liu, Yang Chen, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Acemath: Advancing frontier math reasoning with post-training and reward modeling. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 3993–4015, 2025.

[37] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[38] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 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, 2025. Notion Blog.

[39] Thang Luong, Dawsen Hwang, Hoang H. Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, Alex Zhai, Clara Huiyi Hu, Henryk Michalewski, Jimin Kim, Jeonghyun Ahn, Junhwi Bae, Xingyou Song, Trieu H. Trinh, Quoc V. Le, and Junehyuk Jung. Towards robust mathematical reasoning. In *Proceedings*

of the 2025 Conference on Empirical Methods in Natural Language Processing, 2025. URL https://aclanthology.org/2025.emnlp-main.1794/.

[40] OpenAI. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/, March 2026. Accessed: 2026-05-07.

[41] Jiayi Pan, Xiuyu Li, Long Lian, Charlie Snell, Yifei Zhou, Adam Yala, Trevor Darrell, Kurt Keutzer, and Alane Suhr. Learning adaptive parallel reasoning with language models. *arXiv preprint arXiv:2504.15466*, 2025.

[42] Aneesh Pappu, Mirac Suzgun, Yongchan Kwon, Federico Bianchi, Batu El, Mykel J. Kochenderfer, Hancheng Cao, and James Zou. Self-organizing agent teams learn to reason together, 2026. URL https://arxiv.org/abs/2609.22682.

[43] Jongho Park, Vasilis Kontonis, Shivam Garg, Akshay Krishnamurthy, and Dimitris Papailiopoulos. Scaling discovery through test-time communication, 2026. URL https://arxiv.org/abs/2609.21032.

[44] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

[45] Jianing Qi, Xi Ye, Hao Tang, Zhigang Zhu, and Eunsol Choi. Learning to reason across parallel samples for llm reasoning. *arXiv preprint arXiv:2506.09014*, 2025.

[46] 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. *arXiv preprint arXiv:2311.12022*, 2023.

[47] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[48] 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. *arXiv preprint arXiv: 2409.19256*, 2024.

[49] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

[50] Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, et al. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*, 2025.

[51] The Deep Think Team. Gemini 3 Deep Think: Advancing science, research and engineering. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-deep-think/, February 2026. Accessed: 2026-05-07.

[52] The Gemini Team. Gemini 3.1 Pro: A smarter model for your most complex tasks. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/, February 2026. Accessed: 2026-05-07.

[53] Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024.

[54] Siddarth Venkatraman, Vineet Jain, Sarthak Mittal, Vedant Shah, Johan Obando-Ceron, Yoshua Bengio, Brian R Bartoldson, Bhavya Kailkhura, Guillaume Lajoie, Glen Berseth, et al. Recursive self-aggregation unlocks deep thinking in large language models. *arXiv preprint arXiv:2509.26626*, 2025.

[55] Anna Vettoruzzo, Mohamed-Rafik Bouguelia, Joaquin Vanschoren, Thorsteinn Rögnvaldsson, and KC Santosh. Advances and challenges in meta-learning: A technical review. *IEEE transactions on pattern analysis and machine intelligence*, 46(7):4763–4779, 2024.

[56] Fanqi Wan, Weizhou Shen, Shengyi Liao, Yingcheng Shi, Chenliang Li, Ziyi Yang, Ji Zhang, Fei Huang, Jingren Zhou, and Ming Yan. Qwenlong-l1: Towards long-context large reasoning models with reinforcement learning. *arXiv preprint arXiv:2505.17667*, 2025.

[57] Haonan Wang, Chao Du, Kenji Kawaguchi, and Tianyu Pang. Think in parallel, answer as one: Logit averaging for open-ended reasoning. *arXiv preprint arXiv:2512.02874*, 2025.

[58] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-Agents enhances large language model capabilities, 2024. URL https://arxiv.org/abs/2406.04692.

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

[60] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

[61] Ziqi Wang, Boye Niu, Zipeng Gao, Zhi Zheng, Tong Xu, Linghui Meng, Zhongli Li, Jing Liu, Yilong Chen, Chen Zhu, et al. A survey on parallel reasoning. *arXiv preprint arXiv:2510.12164*, 2025.

[62] Hao Wen, Yifan Su, Feifei Zhang, Yunxin Liu, Yunhao Liu, Ya-Qin Zhang, and Yuanchun Li. Parathinker: Native parallel thinking as a new paradigm to scale llm test-time compute. *arXiv preprint arXiv:2509.04475*, 2025.

[63] Haotian Wu, Bo Xu, Yao Shu, Menglin Yang, and Chengwei Qin. Thinking with nothinking calibration: A new in-context learning paradigm in reasoning large language models. *arXiv preprint arXiv:2508.03363*, 2025.

[64] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.00724.

[65] Ran Xu, Yuchen Zhuang, Zihan Dong, Jonathan Wang, Yue Yu, Joyce C Ho, Linjun Zhang, Haoyu Wang, Wenqi Shi, and Carl Yang. Acesearcher: Bootstrapping reasoning and search for llms via reinforced self-play. *arXiv preprint arXiv:2509.24193*, 2025.

[66] Xiaoyue Xu, Qinyuan Ye, and Xiang Ren. Stress-testing long-context language models with lifelong icl and task haystack. *Advances in Neural Information Processing Systems*, 37:15801–15840, 2024.

[67] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[68] Xinyu Yang, Yuwei An, Hongyi Liu, Tianqi Chen, and Beidi Chen. Multiverse: Your language models secretly decide how to parallelize and merge generation. *arXiv preprint arXiv:2506.09991*, 2025.

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

[70] Yue Yu, Zhengxing Chen, Aston Zhang, Liang Tan, Chenguang Zhu, Richard Yuanzhe Pang, Yundi Qian, Xuewei Wang, Suchin Gururangan, Chao Zhang, Melanie Kambadur, Dhruv Mahajan, and Rui Hou. Self-generated critiques boost reward modeling for language models. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 11499–11514, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.573. URL https://aclanthology.org/2025.naacl-long.573/.

[71] Hao Zhao, Yuejiang Liu, Alexandre Alahi, and Tao Lin. On pitfalls of test-time adaptation. *arXiv preprint arXiv:2306.03536*, 2023.

[72] Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning. *arXiv preprint arXiv:2402.04833*, 2024.

[73] Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. The majority is not always right: Rl training for solution aggregation. *arXiv preprint arXiv:2509.06870*, 2025.

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

[75] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372.

[76] Adam Zweiger, Jyothish Pari, Han Guo, Ekin Akyürek, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. *arXiv preprint arXiv:2506.10943*, 2025.

A Compute, Latency, and Efficiency Analysis

We verify that TTT-Consolidate's gains come from its consolidation mechanism rather than simply more test-time computation. We first give a theoretical complexity analysis of TTT-Consolidate against majority voting and in-context aggregation, then present empirical evidence against compute-matched baselines.

A.1 Theoretical Complexity Analysis

Let K denote the number of intermediate trajectory samples, L the approximate number of tokens in each trajectory, and T the length of the final generated solution. We analyze scaling with K while holding the model, L, and T fixed. Under a token-proportional computation model, sampling K trajectories requires $\Theta(KL)$ total computation (Table 4). Majority voting therefore has a wall-clock advantage when parallel hardware is sufficient: all trajectories can be generated in parallel, while voting cost is negligible. However, this does not make additional trajectories free. Generating K trajectories concurrently still requires $\Theta(KL)$ total model computation. TTT-Consolidate has the same linear asymptotic growth in total computation but a less favorable latency profile, because each consolidation update is sequentially dependent on the updated model state. Its advantage is accuracy per unit of total computation; we do not claim that TTT-Consolidate dominates majority voting on every system dimension, but it is substantially more effective when total computation or serving cost is important, because equal or greater computation spent on additional rollouts does not recover its performance.

Table 4: **Asymptotic scaling of aggregation methods** with the number of sampled trajectories K. TTT-Consolidate matches majority voting's linear compute in K, avoids the quadratic cost of in-context aggregation, and pays only in sequential latency.

Method	**Total compute as K increases**	**Latency, sufficient parallel hardware**
Majority voting	$\Theta(KL)$ sampling + $\Theta(K)$ voting = $\Theta(KL)$	$\Theta(L)$, independent of K with concurrent generation
TTT-Consolidate	$\Theta(KL)$ sampling + $\Theta(KL)$ consolidation + $\Theta(T)$ generation = $\Theta(KL+T)$	Sampling parallelizes; consolidation is sequential, $\Theta(KL)$, followed by $\Theta(T)$
In-context aggregation	$\Theta(KL)$ sampling + $\Theta((KL)^2)$ prefill + $\Theta(TKL+T^2)$ decode = $\Theta(K^2)$ w.r.t. K	Sampling parallelizes; aggregation cost grows with context length

A.2 Empirical Evidence at Matched Compute

Section 5.2 reports TTT-Consolidate's accuracy against majority voting and in-context aggregation at matched compute (Tables 2 and 3). Here we extend that comparison to reward-guided and recursive-refinement baselines, and to inference latency and memory.

We also compare against three additional baselines at matched compute, using TTT-Consolidate @8's budget of 1.050e15 total FLOPs: reward-model Best-of-N [14] using AceMath-7B-RM [36], PRM reranking using Qwen2.5-Math-PRM-7B [34], and Recursive Self-Aggregation [54]. Recursive refinement receives N=8 candidate solutions and runs T=2 rounds; in each round, it draws N=8 subsets of k=4 candidates and generates one new candidate per subset, replacing the population. Table 5 shows that TTT-Consolidate outperforms every baseline at equal compute, both for *None* and *Minority* tiers.

Latency and memory. We measure inference latency using Qwen3-1.7B on a single H100 GPU. All methods receive the same precomputed trajectories, so the latency for sampling trajectories is not counted as part of the total latency since all methods share this measurement. For TTT-Consolidate, "Consolidate" denotes the complete test-time learning stage that consolidates all trajectories through LoRA updates. Table 6 shows that TTT-Consolidate achieves the lowest latency compared to the aggregation baselines, being 2.18× faster than the strongest baseline (25.238s vs. 55.038s for Prompt Aggregation in Thinking mode) while improving *Minority* accuracy by 9.6 points and *None* accuracy by 10.6 points. TTT-Consolidate has higher peak GPU memory because test-time learning requires gradients and optimizer states. However, it uses a substantially smaller KV cache at decode time because the consolidated model generates from the

Table 5: **Iso-compute comparison against reward-guided and recursive-refinement baselines.** All methods run at TTT-Consolidate@8's budget (≈1.05e15 pipeline FLOPs). Best-of-N and PRM reranking select from N=10 sampled solutions. Recursive Refinement runs T=2 rounds, each regenerating N=8 candidates from subsets of k=4. Reranking scores 0.0 on *None* because it can only select an answer that is already in the pool, and every input there is wrong. TTT-Consolidate outperforms the strongest baseline on *Minority* and *None*.

Method	Setting	Pipeline FLOPs	*Minority*	*None*
Best-of-N, AceMath-7B-RM	N=10	1.040e15	24.5	0.0
PRM rerank, Qwen2.5-Math-PRM-7B	N=10, last step	1.040e15	29.1	0.0
Recursive Refinement	N=8, k=4, T=2	1.048e15	17.9	5.4
TTT-Consolidate	K=8	**1.050e15**	**35.4**	**23.6**

Table 6: **Inference runtime and memory usage on a single H100 GPU.** Trajectory sampling is shared by all methods and excluded from runtime. Baselines decode from a ∼6.2K-token prompt of concatenated trajectories. TTT-Consolidate instead consolidates the trajectories into LoRA weights (1.9 s) and decodes from the 156-token problem alone. TTT-Consolidate has the lowest total latency and the smallest KV cache, while achieving the highest accuracy on both tiers. The cost is ∼6 GB higher peak memory for gradients and optimizer states.

				Runtime				Memory		Accuracy	
Method	**Mode**	**Out. budget**	**Prompt**	**Consolidate**	**Decode**	**Total**	**Speed**	**Peak KV**	**Peak mem**	***Minority***	***None***
		tok	*tok*	*s* ↓	*s* ↓	*s* ↓	*ms/tok* ↓	*GB* ↓	*GB* ↓	*%* ↑	*%* ↑
Prompt agg	CoT	8,192	6,182	–	25.867	25.867	3.158	1.649	15.742	17.2	4.0
SFT agg	CoT	8,192	6,182	–	25.867	25.867	3.158	1.649	15.742	10.2	3.2
Prompt agg	Thinking	16,384	6,178	–	55.038	55.038	3.359	2.588	15.761	25.8	13.0
RL agg	Thinking	16,384	6,178	–	55.034	55.034	3.359	2.588	15.742	22.0	3.8
TTT-Consolidate	CoT	8,192	**156**	1.946	23.291	**25.238**	**2.843**	**0.957**	21.962	**35.4**	**23.6**

156-token problem prompt rather than attending to all concatenated trajectories; this also avoids the growth in KV memory incurred by in-context aggregation as the number of trajectories increases. The gain comes from TTT-Consolidate's consolidation mechanism, not more test-time computation. Giving the baselines more compute still does not reproduce TTT-Consolidate's performance.

B Ablation Studies

B.1 Joint Bi-Level Optimization is Necessary

We verify that TTT-Consolidate's improvement arises from its bi-level consolidation mechanism itself, rather than simply from the frontier-model supervision used in its outer loop. SFT Aggregation in CoT mode is trained with the same questions, trajectory groups, and frontier-generated target solutions as TTT-Consolidate (SFT), so both methods receive identical supervised examples and expert supervision. Table 7 compares the two methods on the 10 benchmarks from Table 1: TTT-Consolidate consistently and substantially outperforms direct supervised aggregation. TTT-Consolidate uses bi-level optimization with two nested loops, where the inner loop is optimized by the outer loop. We isolate each loop's contribution in turn.

Table 7: **Comparing different consolidation methods trained with frontier supervision.** SFT Aggregation (CoT) and TTT-Consolidate-SFT train on the same set of identical questions, trajectory groups, and frontier-generated reference solutions. TTT-Consolidate-SFT's positive performance gain improves with model scale. Its improvement therefore comes from the meta-learned test-time training, not simply frontier supervision.

Model	Trajc	SFT Agg	TTT-Consolidate	Gain
Qwen3-1.7B	None	3.2	**23.6**	**+20.4**
Qwen3-1.7B	Minority	10.2	**35.4**	**+25.2**
Qwen3-4B	None	7.7	**40.8**	**+33.1**
Qwen3-4B	Minority	15.4	**57.6**	**+42.2**

Without Inner Loop. If the inner loop is removed, training reduces to either (1) SFT Aggregation, already shown above, or (2) SFT on math problems with a single CoT trace from the frontier teacher

Table 8: **Outer-loop-only ablation.** We remove the inner loop, reducing training to SFT on a single frontier-teacher CoT trace, ignoring sampled trajectories, evaluated with and without aggregation at inference. The outer loop alone is insufficient. SFT on frontier supervision does not match `TTT-Consolidate`'s performance.

	AIME26	Apex25	ArxivMath	HMMT2602
No aggregation at inference				
SFT-1.7B	3.3	0	5.6	3.0
SFT-4B	14.2	0	11.1	6.1
With aggregation at inference				
SFT-1.7B	0.8	0	0.7	1.5
SFT-4B	3.3	0	2.4	3.0
`TTT-Consolidate` (test-time consolidation)				
`TTT-Consolidate`-1.7B	**26.7**	**16.7**	**23.6**	**18.2**
`TTT-Consolidate`-4B	**39.1**	**33.3**	**37.9**	**48.5**

Table 9: **Inner-loop-only ablation.** We start each problem's test-time update from an untrained LoRA initialization, without outer-loop meta-learning. Plain test-time training fails to consolidate trajectories, so both loops under the bilevel setup are needed.

Backbone	Setting	AIME26	Apex25	ArxivMath	HMMT2602
Qwen3-1.7B (None)	Only inner	5.0	0	10.8	3.8
	Inner + outer	**26.7**	**16.7**	**23.6**	**18.2**
Qwen3-1.7B (Minority)	Only inner	6.7	4.2	9.0	4.6
	Inner + outer	**30.0**	**25.0**	**34.7**	**27.3**
Qwen3-4B (None)	Only inner	14.2	0	12.2	13.6
	Inner + outer	**39.1**	**33.3**	**37.9**	**48.5**
Qwen3-4B (Minority)	Only inner	15.0	0	14.9	9.1
	Inner + outer	**47.6**	**41.6**	**42.9**	**57.7**

models (same frontier supervision as `TTT-Consolidate`), where sampled trajectories are ignored. Table 8 reports this scenario for the *None* tier, both with and without aggregation at inference time. `TTT-Consolidate` outperforms both inference settings under this SFT-trained baseline, with strong performance gains.

Without Outer Loop. If the outer loop is removed, meta-learning is removed, and the inner-loop-only ablation applies the inner-loop update at test time from the default, untrained LoRA initialization — this is plain test-time training (TTT) without meta-learning. Table 9 shows that this ablation performs much worse than the complete `TTT-Consolidate` under the same evaluation setting.

Together, these two ablations show that direct outer-loop supervision (i.e., SFT) is insufficient, while inner-loop updates without meta-learning are also ineffective. `TTT-Consolidate`'s gains arise from optimizing the two components jointly in a bi-level optimization framework, so that the outer loop learns how the inner-loop adaptation should consolidate trajectories.

B.2 Isolating the Source of Improvement

We rule out three alternative explanations for `TTT-Consolidate`'s improvement that do not require genuine cross-trajectory consolidation: repeated exposure to the problem, generic task-distribution activation, and gains available from the learned pre-adaptation initialization alone. Table 10 reports matched controls for each. We additionally isolate cross-trajectory interaction directly with a **Single-Trace×8** baseline, where we independently adapt on a single trajectory in the group, obtaining 8 different LoRAs, each generating one final solution, and apply majority voting to determine the final answer. This condition uses all eight trajectories and similar total adaptation compute as full `TTT-Consolidate`, and produces eight final solutions. However, information cannot be shared across trajectories before generating the solutions.

Single-Trace×8 performs much worse than `TTT-Consolidate`, showing that independently encoding each trajectory and voting over the generated answers cannot reproduce `TTT-Consolidate`'s improvement. The other controls also perform far below `TTT-Consolidate`, showing that passive question repetition and task-distribution activation

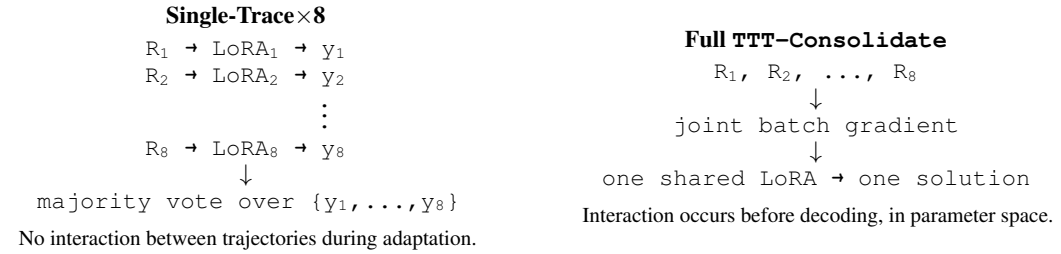


Figure 8: **Isolating where trajectory information can interact: Single-Trace×8 vs. full `TTT-Consolidate`.** Single-Trace×8 adapts one LoRA per trajectory and combines independent answers from the 8 LoRAs only by majority voting. `TTT-Consolidate` aggregates all trajectories into one shared LoRA before decoding a single final solution.

Table 10: **Isolating the source of `TTT-Consolidate`'s improvement.** Every ablation control falls far below full `TTT-Consolidate`. Our method's performance gain requires jointly consolidating the target problem's trajectories during adaptation.

Condition	Inner-loop input	Avg. acc.	Δ pre-adapt.	Δ post-adapt.	Isolate
Pre-adaptation	No inner-loop update	7.5%	–	−14.5	Initialization alone
Passive repetition	8 copies of the target question	5.4%	−2.1	−16.6	Repeated exposure
Task activation	Trajectory groups from other problems, same task distribution	4.0%	−3.5	−18.0	Task-distribution activation w/o target-specific info
Single-Trace×8	One trajectory per LoRA, 8 LoRAs, majority vote	11.7%	+4.2	−10.3	Absence of cross-trajectory interaction
`TTT-Consolidate`	All 8 target trajectories, jointly	**22.0%**	+14.5	**–**	**–**

do not contribute to `TTT-Consolidate`'s improvement. (To make the 8× Single-Trace baseline a fair comparison to `TTT-Consolidate`, we additionally sample 8 CoTs from the adapted model after the inner loop and perform a majority vote of these, which yields a ∼1% average additional improvement.)

`TTT-Consolidate` requires information from multiple target-specific trajectories to be jointly consolidated in the same adapted LoRA before it generates the final solution. We accordingly scope the weak-to-strong generalization claim to: **`TTT-Consolidate` can recover from unsuccessful trajectories and arrive at an improved solution through aggregation as test-time learning.**

B.3 Role of Reinforcement Learning in Consolidation

Because RL jointly optimizes the LoRA initialization *and* the model's reasoning after inner-loop updates, the effects of initialization and consolidation are coupled in the post-adaptation measurement. To directly separate performance available in the initialization from performance obtained after trajectory consolidation, we evaluate the same RL-enhanced `TTT-Consolidate` model with and without consolidation at inference time (Table 11).

The only difference between the two rows within each checkpoint is whether the inference includes the inner-loop adaptation. The inner-loop adaptation clearly improves every benchmark, so the final `TTT-Consolidate` model does not obtain its performance from the initialization alone; its test-time updates contribute a substantial improvement. `TTT-Consolidate`-RL is initialized from the `TTT-Consolidate`-SFT policy before GRPO training, following prior art, so it does inherit frontier supervision through this initialization. To isolate RL's own contribution, we additionally evaluate a compute- and initialization-matched RL Aggregation baseline that initializes from SFT Aggregation and applies the same GRPO training, verifiable reward, rollout budget, and RL update budget used for `TTT-Consolidate`-RL (Table 12).

This comparison controls for both frontier-supervised initialization and RL training; `TTT-Consolidate` still outperforms this baseline under the matched SFT-to-RL learning path. We do not interpret this as showing that RL primarily improves the consolidation process (though this is benchmark-dependent); however, the adaptation gain under the same checkpoint shows that the inner loop remains important. RL primarily strengthens the initialization, but the resulting model

Table 11: **`TTT-Consolidate` model with and without test-time adaptation** on four *None* tier benchmarks. Inner-loop adaptation improves both the SFT and RL models on every benchmark. Performance gains do not come from the initialization alone.

		AIME26	Apex25	ArxivMath	HMMT2602	GPQA
TTT-Consolidate-SFT	Before TTT	16.7	0	2.8	6.1	6.6
	After TTT	26.7	16.7	23.6	18.2	22.7
	Gain from adaptation	+10.0	+16.7	+20.8	+12.1	+16.1
TTT-Consolidate-RL	Before TTT	16.7	0	4.2	9.1	19.7
	After TTT	23.5	16.7	26.0	21.7	40.0
	Gain from TTT	+6.8	+16.7	+21.8	+12.6	+20.3

Table 12: **Matched SFT→GRPO training path for RL Aggregation and `TTT-Consolidate`.** Both methods get the same frontier-supervised initialization and GRPO budget. RL Aggregation stays behind; thus, RL alone does not explain `TTT-Consolidate`'s improvement.

Method	AIME26	ArXivMath	Apex25	HMMT-2602	Avg
RL Aggregation: Base → GRPO (base-initialized)	6.7	2.8	0.0	0.4	2.48
RL Aggregation: SFT → GRPO (matched)	10.0	1.2	0.0	9.1	5.08
TTT-Consolidate-SFT → GRPO (TTT-Consolidate-RL)	**23.5**	**26.0**	**16.7**	**21.7**	**21.98**

still relies substantially on trajectory-conditioned test-time adaptation: *RL improves the final adapted policy's ability to reason over consolidated trajectories, while its current training gains arise mainly from a stronger initialization for consolidation.*

C Consolidation Dataset: Construction, Quality Control, and Verification

C.1 Evaluation Benchmark and Trajectory Preparation

Benchmark Coverage and Question Pool We assemble a pool of 1,645 unique questions drawn from ten challenging reasoning benchmarks spanning competition mathematics and scientific reasoning. We detail the full pipeline to construct the reasoning consolidation dataset used in our experiments. Table 13 provides a complete breakdown of each benchmark, its domain, and the number of questions it contributes. Together, these benchmarks cover a broad difficulty spectrum: from competition-level problems designed for elite high-school contestants (AIME, HMMT, CMIMC) and olympiad-level challenges (IMO-Bench) to research-level mathematics (ArXivMath), PhD-level scientific reasoning across physics, chemistry, and biology (GPQA Diamond), and multi-domain STEM theorem application (TheoremQA). Notably, the difficulty of these benchmarks substantially exceeds the reach of smaller models, posing a genuine challenge even for capable reasoning models.

Table 13: **Summary of the ten reasoning benchmarks.** Each question is used to generate 40 candidate trajectories (5 source models × 8 samples), from which three quality-stratified trajectory groups are formed.

Benchmark	Domain	# Questions
Math Reasoning		
AIME 2025	Competition Math	30
AIME 2026	Competition Math	30
Apex 2025	Competition Math	12
ArXiv Math	Research Math	72
CMIMC 2025	Competition Math	40
HMMT Feb 2026	Competition Math	33
HMMT Nov 2025	Competition Math	30
IMO-Bench	Olympiad Math	400
STEM Reasoning		
GPQA Diamond	Scientific Reasoning	198
TheoremQA	STEM Theorem Reasoning	800
Total		**1,645**

Source Models and Trajectory Generation For each question in the pool, we sample 8 independent solution attempts from each of five source models, yielding 40 candidate trajectories per question. The five source models span a range of scales and capabilities. **Qwen2.5-1.5B-Instruct**, **Qwen3-1.7B**, and **Qwen3-4B** represent lightweight to mid-range open models. **Qwen3-30B-A3B** is

a sparse mixture-of-experts model with 30B total but only 3B active parameters per token, offering strong reasoning at moderate inference cost. **Gemini 2.0 Flash** is a capable closed-source model serving as the strongest single contributor. The first four open-source models operate in extended thinking mode and produce very long internal reasoning traces before writing a final answer. Gemini 2.0 Flash is a closed-source model that outputs only a brief CoT solution without an explicit reasoning trace.

CoT Extraction from Thinking Outputs For the four open-source thinking models, each raw generation contains a lengthy internal reasoning trace in `<think>...</think>` tags, which can exceed tens of thousands of tokens and is therefore too long to include directly in aggregation prompts. We extract only the brief final solution appearing after the thinking block as the CoT trajectory. For Gemini 2.0 Flash, the full output is used directly. As shown in Figure 9 (left), the extracted CoTs have a median length of 762 tokens (25th–75th percentile: 559–927), confirming they are compact enough for multi-trajectory aggregation contexts.

Quality-Based Stratification For each question, we sample 8 trajectories from the 40 candidates and assign them to one of three quality tiers based on how many of the 8 are correct:

- *None* ($0/8$ correct): all selected trajectories are incorrect, representing the hardest aggregation scenario where the model must extract useful reasoning steps from entirely wrong solutions.
- *Minority* (0–$3/8$ correct): a mix of correct and incorrect trajectories, reflecting the most common real-world setting where the aggregator must identify and leverage correct solutions among a noisy mixture.
- *Majority* (4–$7/8$ correct): the majority of selected trajectories are correct, an extreme case testing whether the model can reliably synthesize a consistent final answer.

Two boundary cases arise in practice: some questions are so easy that all 40 candidates are correct (pass@40 $= 100\%$), and others are so hard that all 40 are wrong (pass@40 $= 0\%$). For these questions, we use frontier models (GPT-5.3 and Claude 4.6) to manually correct the imbalance with access to the ground-truth answer. In the all-correct case, a subset of trajectories is rewritten to introduce plausible reasoning errors that lead to a wrong final answer. In the all-wrong case, a subset of trajectories is corrected to produce a valid solution. This ensures that every question participates in all three quality tiers, yielding $1{,}645 \times 3 = 4{,}935$ trajectory groups in total.

Source Model Composition across Quality Tiers As shown in Figure 9 (right), source model composition shifts systematically across quality tiers. In the *None* tier, 51% of trajectories come from Qwen2.5-1.5B, the weakest model, because it fails most often. As quality increases, stronger models contribute more: Gemini 2.0 Flash rises from 5% (*None*) to 30% (*Majority*), and Qwen3-30B-A3B from 9% to 17%, while Qwen2.5-1.5B's share falls to 20%. This emergent shift confirms that model capability and trajectory quality are tightly coupled.

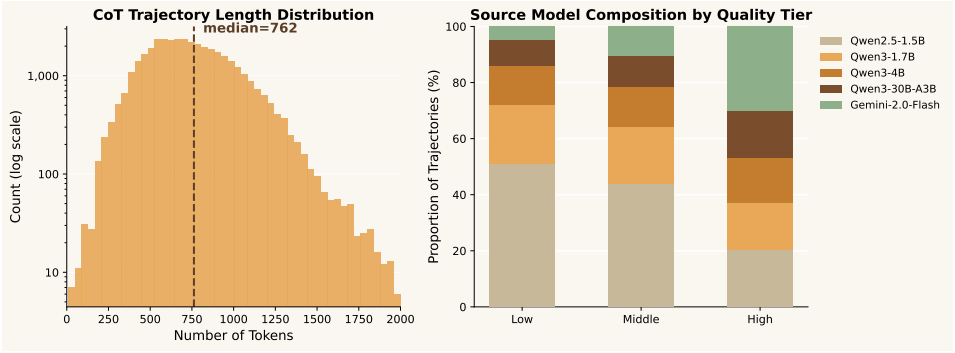


Figure 9: **Left**: Distribution of extracted CoT lengths over reasoning trajectory collection. **Right**: Source model composition by quality tier, where weaker models dominate the *None* tier while stronger models contribute more in the *Majority* tier.

C.2 *OpenParallelThinking* Training Corpus Construction

This section details the construction pipeline for *OpenParallelThinking* summarized in Section 3.

Source Diversity We pool problems from OmniMath [23], DeepScale [38], NuminaMath [33], and OpenMathInstruct [53], yielding 19,615 unique problems after hash-based deduplication.

Trajectory Diversity We sample from four solvers spanning different scales, families, and training recipes: R1-7B [16], Nemotron3-14B [11], Qwen3-4B, and Qwen3-30B-A3B [67], since a homogeneous source would bias the aggregator toward a single reasoning style. We drop the extended reasoning segment of each trajectory with probability 0.5[3], exposing the aggregator to a realistic mix of detailed and compact reasoning. Trajectories are then randomly grouped per question under a 16K-token budget, so that group size varies naturally with length [72] (Figure 2, left).

Balanced Difficulty We filter trajectory groups by correctness ratio (Figure 2, center) to provide more training examples on reasoning aggregation over groups where the majority are incorrect trajectories, which provides stronger learning signals without the majority correct answer shortcut.

Final Dataset The final release contains 12,810 questions and 63,507 trajectory groups. We sample 8,000 instances for RL training and reserve the rest for SFT.

C.3 Verifying Genuine Trajectory Consolidation

A natural concern is whether `TTT-Consolidate`'s inner loop learns to memorize input trajectories rather than consolidate useful sub-reasoning, and whether its gains on the *None* tier are an artifact of the frontier-model-rewritten trajectories used to balance that tier (Section C.1) rather than genuine consolidation of naturally occurring failures. We address both concerns with a construction-based argument and a new LLM-as-judge analysis.

Ruling out memorization by construction. When all 8 trajectories in the *None* tier are naturally unsuccessful trajectories sampled from source models, each ending in a wrong answer, there are no correct candidates to memorize and recall. Averaged over the ten benchmarks, `TTT-Consolidate` reaches 23.58 vs. 13.02 for the strongest baseline (Qwen3-1.7B) and 40.83 vs. 19.63 (Qwen3-4B). For `TTT-Consolidate` to output a correct answer in this setting, it must not have only memorized trajectories leading to the correct answer. Figure 4 also shows that `TTT-Consolidate` generally improves as more incorrect trajectories are added, while baselines stagnate or degrade. Thus, `TTT-Consolidate` does not depend on simply copying the input trajectories' answers.

Ruling out trajectory-editing artifacts. Edited trajectories are a negligible presence in the *None* setting. Among all 1,645 evaluation questions, only two (one from GPQA; one from TheoremQA) contain any rewritten trajectory in the *None* setting; the remaining eight benchmarks contain none. Counting these two questions as incorrect decreases the GPQA and TheoremQA scores by less than 0.5 percentage points and leaves every other benchmark score unchanged. The weak-to-strong gains in Table 1 are therefore primarily driven by naturally occurring model failures, not edited rewriting.

Judge Setup We use three frontier LLM judges, Fable 5, Opus 5, and GPT-5.6-Sol, each run at extra-high reasoning effort. Every sampled solution is independently labeled by all three judges, and the final label is the majority vote across the three.

Synthesis and Artifact-Exploitation Judging (*None*) We sample 100 `TTT-Consolidate` (Qwen3-1.7B) solutions on *None*-tier problems, where every one of the eight input trajectories reaches an incorrect answer. For each solution, the judges are shown the problem, the eight input trajectories, and `TTT-Consolidate`'s final solution, and are asked two binary questions: (1) whether the final solution derives an improved reasoning trajectory from the incorrect inputs (*synthesis*) rather than simply copying one input's reasoning verbatim, and (2) whether the solution exploits stylistic or distributional cues rather than demonstrating mathematical understanding (*artifact exploitation*). 98/100 solutions are labeled as synthesis, and 97/100 show no evidence of artifact exploitation. We additionally audit the two *None*-tier questions containing an edited trajectory (Section C.1) specifically

[3]For thinking models, the reasoning segment corresponds to the content between two special thinking tokens, most commonly `<think>` and `</think>`.

Table 14: Prompt template for the LLM-as-judge analysis on the final solutions produced by `TTT-Consolidate`.

Component	Content
Judge Prompt	You are evaluating whether a model's final solution to a problem reflects {synthesis vs. artifact exploitation | consolidation vs. memorization}. Problem: {problem} Input trajectories: {trajectory$_0$}, ..., {trajectory$_7$} Final solution: {solution} Question: {task-specific question} Respond with a single label from {label set} and a one-sentence justification.

for exploitable artifacts; the judges find that the edits introduce mathematically meaningful errors without creating clear exploitable cues.

Memorization Judging (*Minority*) We separately sample 100 `TTT-Consolidate` (Qwen3-1.7B) solutions on *Minority*-tier problems, where a subset of the eight input trajectories is already correct. Judges are shown the same problem, trajectory, and solution triple, and asked to label the final solution as *memorization* (near-verbatim reproduction of one input trajectory's reasoning and answer) or *consolidation* (reasoning synthesized or corrected using information across multiple inputs). 1/100 solutions is labeled memorization.

Judge Prompt Template Table 14 shows the prompt template shared across the three judging tasks, with the task-specific question and label set substituted for each protocol above.[4]

Taken together, these results show clear evidence that `TTT-Consolidate` consolidates multiple naturally sampled unsuccessful trajectories into a stronger, improved solution, rather than memorizing correct answers or exploiting artifacts introduced by trajectory editing.

D Training and Inference Details

Hardware Information All experiments are conducted on a single NVIDIA H100 GPU with 93 GB of HBM on an internal Slurm-managed research cluster. Each job is assigned 64 CPU cores and 64 GB of system RAM with exclusive GPU access throughout training.

Baseline Training Hyperparameters The SFT aggregation model is obtained by full-parameter fine-tuning using LLaMA-Factory [75]. We adopt the no-thinking chat template with a context window of 20,576 tokens. Training runs for 1 epoch with a learning rate of 1×10^{-5}, a cosine schedule with 10% linear warmup, and an effective global batch size of 32. All training is conducted in `bf16` precision. For RL aggregation, we implement GRPO [47] under the Verl [48] framework, with thinking mode enabled. We use a learning rate of 3×10^{-6}, a global batch size of 64 prompts, and $n = 5$ rollouts per prompt. The maximum prompt and response lengths are set to 8,192 and 16,384 tokens, respectively.

Our Method Training Hyperparameters We train `TTT-Consolidate` for up to 1 epoch with early stopping based on validation loss, using PyTorch Lightning[5] modified to support meta-gradients. The outer loop optimizer is AdamW [37] with a learning rate of 1×10^{-5}, weight decay of 0.01, and a cosine schedule with 3% warmup steps. The consolidation module is parameterized as a LoRA adapter with rank 256 and $\alpha = 128$, applied to all weight matrices of the model. For inner-loop adaptation, we reuse the encoding scheme and hyperparameters from that prior work, including its Truncated Gradient Unrolling (TGU) scheme [44]: the first $S - T$ inner steps are executed without graph retention, and only the final T steps are differentiated, reducing memory from $O(S|\phi|)$ to $O(T|\phi|)$.

[4] More detailed instructions, including definitions of artifact exploitation, memorization, and consolidation, will be released with the code.

[5] https://lightning.ai/docs/pytorch/stable/

Table 15: Prompt templates for model-based aggregation baselines and `TTT-Consolidate`.

Method	Prompt Template
Prompt Aggregation SFT Aggregation RL Aggregation	Given the following problem: {problem} and these solution attempts: Solution attempt 0: {trajectory$_0$} Solution attempt 1: {trajectory$_1$} … (more attempts) Carefully review the provided solutions, using them as starting points, correcting mistakes, filling in gaps, and/or combining useful ideas, to produce a final, comprehensive, and correct solution to the problem. Analyze step by step and put your final answer in the format \boxed{}
`TTT-Consolidate` (Inner Loop)	record$_0$: {trajectory$_0$} record$_1$: {trajectory$_1$} … (one sequence per trajectory, K steps total)
`TTT-Consolidate` (Outer Loop)	record$_0$ to record$_{K-1}$ are the information you memorized. Question: {problem} Please solve this problem step by step by reasoning from your memory and put your final answer in the format \boxed{}. \nAnswer:

For the RL-enhanced variant, we introduce GRPO as the outer-loop objective on top of the existing PyTorch Lightning meta-training framework, without adopting dedicated RL infrastructure such as vLLM or VERL. The outer-loop learning rate is reduced to 1×10^{-6} with weight decay 0.003. Rollouts are capped at 2,048 new tokens. To further reduce peak memory from higher-order gradient computation, we set `unroll_start`$= 3$ with $n_{\text{inner}} = 4$, retaining only the final inner step in the gradient graph.

Evaluation Hyperparameters All methods use greedy decoding (temperature $= 0$), with the end-of-sequence token taken from the model tokenizer's default. The maximum generation length is 8,192 tokens in CoT mode and 16,384 tokens in thinking mode. Baseline inference is accelerated with vLLM [30], while `TTT-Consolidate` generates answers through the Hugging Face Transformers pipeline[6].

E Prompts

The three model-based aggregation baselines (**Prompt Aggregation**, **SFT Aggregation**, and **RL Aggregation**) share an identical prompt template in training and inference, differing only in the underlying model checkpoint. Prompt Aggregation uses the base model directly, while SFT and RL Aggregation use models further fine-tuned via supervised fine-tuning and reinforcement learning, respectively. The shared prompt concatenates all K solution attempts with indexed headers and instructs the model to synthesize a final correct answer by reviewing them. `TTT-Consolidate` uses two distinct prompts. The inner-loop prompt encodes each reasoning trajectory as a named memory record (`record_`i) using a causal language modeling objective, compressing the trajectory set into the LoRA adapter weights via gradient updates. The outer-loop prompt then instructs the adapted model to leverage memorized records to answer questions. Table 15 shows the detailed prompt templates.

F Limitations

Memory Cost of Meta-RL Training. Meta-learning requires backpropagating through the inner optimization loop, which increases memory cost compared to standard fine-tuning. Replacing the SFT outer loss with GRPO further amplifies peak memory usage due to rollout generation and higher-order gradient computation. LoRA substantially reduces this overhead, while we believe further reductions through gradient checkpointing or more aggressive truncation schemes would broaden accessibility to researchers with tighter hardware budgets and enable scaling to larger backbone models.

Scope of Reasoning Domains. Our experiments cover mathematical and STEM reasoning, where trajectory quality can be easily and completely assessed through verifiable final answers. Extending

[6] https://huggingface.co/docs/transformers/v4.43.4/en/main_classes/pipelines

TTT-Consolidate to agentic settings, where reasoning involves tool use, multi-step environment interaction, and open-ended planning, introduces new challenges in defining trajectory quality and designing appropriate reward signals. These are natural and compelling directions for future work, and the meta-learning formulation of TTT-Consolidate is in principle compatible with such settings.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Main content and contributions are introduced in the abstract and introduction

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Limitations are discussed in the Appendix F.

   Guidelines:

   - The answer [N/A] 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: [N/A]

   Justification: The paper does not present any theoretical results.

   Guidelines:

   - The answer [N/A] 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: All information should be mentioned in the Section 3, Section 4 of the main part and in the Appendix (especially the Training and Inference Details).

   Guidelines:

   - The answer [N/A] 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: Samples of the training and evaluation data for our proposed dataset are included in the supplementary. The full dataset and pipeline will be released for camera-ready.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Important information discussed in the Section 4, and details are in the appendix.

   Guidelines:

   - The answer [N/A] 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: For all small benchmark datasets (less than 100 samples), we run them 8 times individually and report their average scores.

   Guidelines:

   - The answer [N/A] 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: Described in detail in Appendix B.

   Guidelines:

   - The answer [N/A] 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: We followed the code of ethics.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: Our work focuses on a neutral technical innovation and does not have clear, direct positive or negative societal impacts at this stage.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work uses open-source models and publicly available datasets, and focuses on relatively neutral Math and STEM reasoning tasks that do not pose significant misuse risks.

    Guidelines:

    - The answer [N/A] 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 datasets used in this work are open-source or publicly available. We properly cite the original sources and follow their respective licenses and terms of use.

    Guidelines:

    - The answer [N/A] 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: The main details of the assets are already provided in Section 3, Section 4, and the Appendix. For future open-source release, we will ensure that comprehensive documentation is provided alongside the released checkpoints and datasets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: In this paper LLMs are only used for editing (e.g., grammar, spelling, word choice), so the core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.