LACE: Lattice Attention for Cross-Thread Exploration in LLMs

Yang Li
Department of Computer Science
Rutgers University

Zirui Zhang
Department of Computer Science
Rutgers University

Yang Liu
Amazon AGI

Chengzhi Mao
Department of Computer Science
Rutgers University

Abstract

Current large language models reason in isolation. Although it is common to sample multiple reasoning paths in parallel, these trajectories do not interact, and often fail in the same redundant ways. We introduce LACE, a framework that transforms reasoning from a collection of independent trials into a coordinated, parallel process. By repurposing the model architecture to enable cross-thread attention, LACE allows concurrent reasoning paths to share intermediate insights and correct one another during inference. A central challenge is the absence of natural training data that exhibits such collaborative behavior. We address this gap with a synthetic data pipeline that explicitly teaches models to communicate and error-correct across threads. Experiments on mathematical reasoning and interactive agent tasks show improvements in candidate coverage and self-selection over independent sampling. Our results suggest that large language models can be more effective when parallel reasoning paths are allowed to interact.

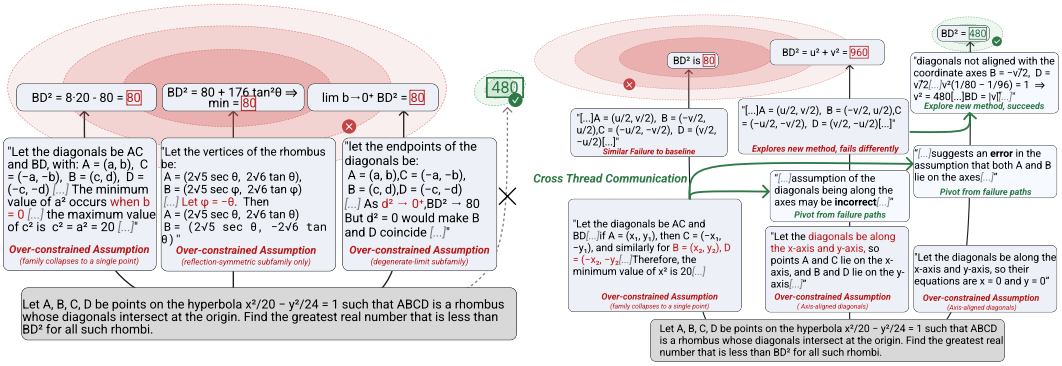
1 Introduction

Large language models achieve strong performance across a range of reasoning tasks, yet they remain constrained by strictly sequential, left-to-right decoding [35]. When tackling difficult proofs or complex plans, standard generation models commit to a single path in a vacuum. These constraints limit the effectiveness of language models in highly complex problem-solving scenarios, underscoring the gap between current decoding algorithms and human-level abilities, which rely on collateral thinking to explore multiple hypotheses at once [17, 16, 18].

A large body of work has studied how to scale test-time compute such that models can solve complex reasoning tasks. Parallel sampling and its variants [5, 56, 42], including generating k independent solutions and selecting the best one post-hoc, significantly improve reasoning performance. However, while existing methods focus on increasing the number of samples, they are burdened because they isolate each thread like k agents solving a puzzle in separate rooms. Independent sampling cannot adapt to the breakthroughs or dead ends of other threads during generation, causing computation to frequently collapse into redundant, correlated errors [19, 22].

In this paper, we introduce an approach for *connecting* parallel generation processes, allowing us to formulate a decoding strategy that adapts and collaborates across threads. Our approach, LACE (Lattice Attention for Cross-thread Exploration), enables threads to share in-situ insights to solve problems collaboratively. Figure 1 shows how our method explores diverse solutions compared to isolated sampling. However, multi-threaded reasoning is more challenging to produce than standard sampling because parallel, logically diverse traces are scarce in standard pre-training corpora.

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



(a) Isolated samples fall into correlated failure modes. **(b)** LACE explores diverse solutions.

Figure 1: **Overcoming correlated failures in reasoning spaces.** For complex problems, correct reasoning paths are often rare and widely separated. **(a) Isolated Sampling:** Standard parallel generation methods independently explore the space, frequently collapsing into **correlated failure modes** driven by shared, over-constrained assumptions. **(b) LACE (Ours):** By introducing **latent cross-thread communication** during generation, our method enables concurrent threads to actively pivot away from known failure paths. This shared context encourages the exploration of diverse reasoning modes, significantly increasing the probability of discovering the correct solution.

Our key insight is that multi-thread communication can occur directly within the latent space to be efficient and effective. Rather than relying on isolated sequence processing or slow text-based coordination, we design and inject explicit parallel attention pathways, expanding the architecture into a 2-D Lattice Attention structure. This introduces a width dimension that actively shares continuous representations across trajectories. We found that by curating synthetic data with explicit interaction points, we can capitalize on our lattice attention layers to teach models how to collaborate. By using Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL) to train this structure, the model explicitly diversifies search strategies and prevents redundant failures, providing a much more efficient search than post-hoc verification.

A key advantage of our framework is that it transforms reasoning from a set of isolated, independent events into a unified, collaborative exploration. Moreover, because this coordination occurs within the latent space, our approach can perform in-situ evaluation to identify and select the optimal solution on the fly, avoiding an additional text-based communication or selection stage. Furthermore, this multi-threaded formulation allows us to seamlessly apply diversity rewards directly during RL, incentivizing the threads to explore distinct reasoning paths and maximize overall accuracy.

Visualizations and empirical experiments show that our collateral reasoning strategy significantly improves reasoning diversity and problem-solving capabilities for several established benchmarks. Our method outperforms independent sampling baselines by a large margin across challenging reasoning datasets including AIME 25 [54], AIME 24 [53], and LiveBench [40]. In addition, our empirical results demonstrate that our method learns to share insights across threads, allowing it to pick the best solution on the fly and dramatically reduce wasted compute.

2 Related Work

External Test-Time Search and Scaling. The paradigm of large language models is shifting from scaling parameters to scaling inference-time compute [31, 43], with recent studies showing that additional reasoning-time computation, often framed as *System 2 thinking*, can yield gains comparable to scaling model size [21]. Existing methods largely realize this extra compute through external search or orchestration, including sequential reasoning strategies such as Chain-of-Thought and its variants [38, 20, 57], structured search methods such as Tree-of-Thoughts [48], and parallel sampling pipelines based on self-consistency [36], bootstrapped Best-of-N aggregation [26], learned verifiers or reward models [32, 5], RL-based parallel reasoning frameworks such as Parallel-R1 and Native Parallel Reasoner [56, 42], and width-oriented external parallel thinking such as ParaThinker [39].

Despite their differences, these approaches coordinate reasoning primarily outside the model's native token-level generation process, and remain vulnerable to biased or correlated failures across independently sampled trajectories [15, 14, 22].

Generation-Time Parallel Reasoning. More recent work has begun to move parallel reasoning into the generation process itself. ParaDecodeOneSeq [50] packs multiple branches into a single sequence for efficiency-oriented decoding. Hogwild! [27] enables *explicit* concurrent attention through shared cross-thread KV states, while GroupThink [13] studies token-level collaboration among concurrent reasoning agents via direct cross-thread interaction. While LACE shares the goal of generation-time cross-thread interaction with these methods, it differs in mechanism. Rather than exposing each thread to token histories of other threads through shared KV states and non-standard masks, it introduces *implicit* cross-thread interaction through a lightweight gated side path over standard attention-derived representations. This preserves the standard causal-attention backbone while allowing threads to influence one another during generation, turning independent sampling into a collaborative process that explicitly targets redundant exploration and correlated errors [19, 12].

3 Method

The central idea of LACE is to make parallel reasoning more exploratory by allowing threads to collaborate while they are generated. For difficult reasoning problems, correct solutions can occupy sparse modes in the solution space. Independent parallel sampling often revisits nearby failure modes, so simply increasing the number of samples may not increase useful coverage. We therefore build LACE around three connected requirements. First, the model needs training data in which different threads explore distinct but comparable solution paths. Second, the model needs an architectural channel that lets these threads exchange information during generation. Third, post-training should strengthen the communication and self-selection behaviors that turn diverse candidates into a better final answer. Following this logic, we first seed multi-thread data for complementary exploration in Section 3.1, then connect threads with Lattice Attention in Section 3.2, and finally post-train the model for collaborative search in Section 3.3.

3.1 Seeding Multi-Thread Data for Complementary Exploration

Collaborative reasoning is only meaningful when parallel threads have something useful to share. The training data must therefore satisfy two properties. First, **reasoning diversity**: each problem should admit multiple logically distinct solution paths, so threads cover different regions of the solution space rather than producing superficial rephrasings. Second, **cross-thread comparability**: the threads should solve the same problem in a shared format, so the model can compare partial progress, recognize successful paths, and avoid repeating known failures. Driven by these criteria, we synthesize multi-thread training data through the pipeline shown in Figure 2.

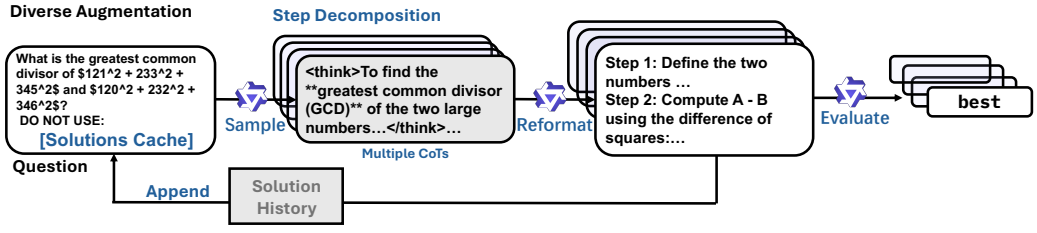


Figure 2: Data curation pipeline for seeding complementary reasoning threads.

Model-Specific Filtering. Not all problems are suitable for multi-thread training. Problems at the extremes of difficulty naturally fail to satisfy our criteria. Overly simple problems typically admit a single obvious solution, which the model tends to sample repeatedly with limited diversity. Conversely, excessively difficult problems prevent the model from sampling any correct solution, thereby lacking positive supervision. Ideal training data for a specific base model requires a balanced mixture of successful and failed reasoning paths [2], since a failed exploration strategy in one thread can still provide useful information to another. We therefore retain only problems with a success-rate

consistency score $S = 1 - \Delta S \in [0, 0.5]$ where ΔS denotes the success-rate difference between naive and diverse sampling. This criterion focuses training on problems where exploration has the potential to uncover correct solutions, making cross-thread recovery and comparison useful.

Diverse Thread Augmentation. To ensure reasoning diversity across threads, we employ an iterative sampling strategy that explicitly discourages redundant solution paths, inspired by DeRL [1]. For each problem, we maintain a *solution history* that accumulates compressed summaries of previously generated reasoning traces. When sampling a new thread, we prepend this history to the prompt with an explicit instruction to avoid the catalogued approaches, for example `DO NOT USE: [Solutions Cache]`. This steers the model toward unexplored strategies and implements a soft rejection sampling procedure over the space of solution methods. Each newly sampled trace is then summarized and appended to the solution history, progressively expanding the set of forbidden paths for subsequent threads. We repeat this process until we obtain T diverse reasoning trajectories per problem. To control sequence length for efficient training, we further apply step decomposition that compresses verbose traces ($>$8000 tokens) into concise step-by-step formats ($<$3000 tokens) while preserving the core logical flow.

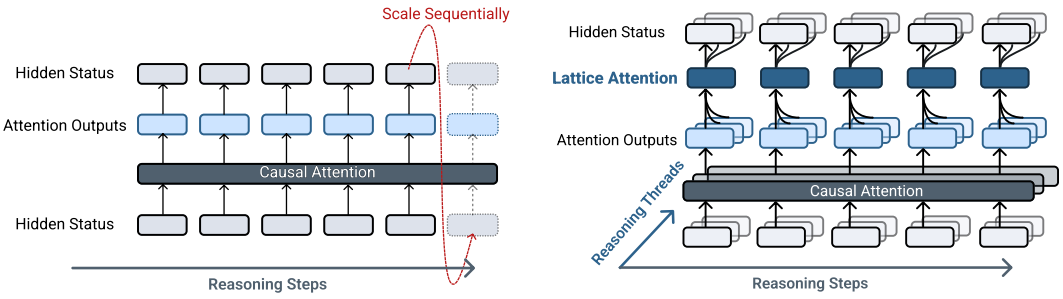
Self-Selection Formatting. To make diversity useful for collaboration, we design a self-selection task that requires cross-thread comparison. Given T parallel reasoning paths, the model must identify the thread yielding the best solution. This task encourages the model to use peer information rather than rely on isolated single-thread reasoning. To construct supervision, we use LLM-as-a-judge [55] to evaluate each thread's validity and logical soundness within the multi-thread context, providing a critique and assigning a grade. The final synthetic training data is formatted as:

[question]
$_t$: [exploration] [comment] [TAG], $t = 1, \ldots, T$

where t denotes the thread index, *exploration* contains the reasoning trace, *comment* holds the judge's comparative evaluation, and *tag* indicates a categorical outcome label. The tag is drawn from BEST, SUCCESS, and FAIL, with exactly one thread designated as BEST. Both the comment and tag are only well-defined through cross-thread comparison, directly training LACE to attend to peer threads without an extra LLM-based selection stage.

3.2 Connecting Threads with Lattice Attention

Given diverse and comparable threads, the next challenge is to let them communicate during generation. We introduce Lattice Attention as a lightweight side path that extends standard causal processing with cross-thread information flow. The mechanism is designed to preserve the pre-trained causal backbone while adding a controlled channel for peer context. Let B denote the batch size, N the number of threads, L the context length, and D the hidden dimension. We denote the head dimension as d and use subscript p for lattice attention components. An overview is shown in Figure 3.



(a) Baseline LLM (b) LACE (ours)

Figure 3: Comparison between the standard parallel decoding pipeline and LACE (ours). Left: a baseline LLM generates reasoning threads independently, so intermediate progress cannot be shared across threads during decoding. Right: LACE (ours) augments the backbone with lattice attention, enabling cross-thread communication through a gated residual path while preserving causal generation within each thread.

Context-Aware Design. A naive design would apply cross-thread attention directly over raw token embeddings or early hidden states. However, reasoning is inherently contextual: whether a token is useful to another thread depends on the partial derivation that precedes it, not just on the token itself. Tokens at aligned positions across threads may therefore be poorly matched before each thread has formed enough local reasoning context. This makes naive cross-thread attention both noisy and hard to train. It also increases computation if attention is extended over all thread-token pairs, and large extra layers trained from scratch may disturb well-learned causal attention layers, especially with limited collaborative data. We therefore use a **context-aware** and **parameter-efficient** design. Instead of attending over raw embeddings, we operate on the output of standard scaled dot-product attention (SDPA), $\mathbf{A}_{\text{std}} \in \mathbb{R}^{(BN)\times L\times D}$, which already encodes each thread's causal context. This lets LACE exchange richer intermediate reasoning states while avoiding redundant computation.

To keep the added module lightweight, we use three strategies. First, SDPA outputs are projected to a lower-dimensional lattice space before cross-thread attention. Second, inspired by ControlNet [51], we selectively plug lattice attention layers into middle-to-late layers of the base model, where cross-thread communication is most useful for complex reasoning [45]. Third, a learned gate modulates the contribution of the lattice path, allowing the model to balance thread-independent and thread-aware processing. Together, these choices limit additional parameters to less than 1% of the original model while enabling cross-thread information exchange with minimal disruption to the pre-trained causal layers.

Cross-Thread Attention. We project the standard attention output into a low-dimensional lattice space, $\mathbf{Z} = \mathbf{A}_{\text{std}}\mathbf{W}_{\text{down},p}$, and compute parallel queries, keys, and values, $\mathbf{Q}_p, \mathbf{K}_p, \mathbf{V}_p = \text{Proj}(\mathbf{Z})$. To encode both token and thread indices, we apply 2D RoPE [33], yielding $\tilde{\mathbf{Q}}_p, \tilde{\mathbf{K}}_p = \text{RoPE}_{\text{2D}}(\mathbf{Q}_p, \mathbf{K}_p; l, n)$, where the first d_t head dimensions encode token position l and the remaining $d_b = d - d_t$ dimensions encode thread index n. Cross-thread attention is applied only among SDPA representations at the same decoding step, so a token at position l can exchange information with peer threads at position l after each thread has already summarized its own causal prefix through SDPA. It does not attend to raw future tokens from other threads. We then group the N same-step representations within each batch item and perform cross-thread attention, $\mathbf{A}_p = \text{Attention}(\tilde{\mathbf{Q}}_p, \tilde{\mathbf{K}}_p, \mathbf{V}_p)$. Following Gated Attention [24, 8, 30], we fuse the standard and lattice paths with a learned gate, $\mathbf{G} = \sigma\left(\text{Linear}([\mathbf{H}_{\text{down}}; \mathbf{A}_p])\right)$ and $\mathbf{O} = \mathbf{A}_{\text{std}}\mathbf{W}_O + (\mathbf{G} \odot \mathbf{A}_p)\mathbf{W}_{O,p}$, where $\mathbf{H}_{\text{down}} = \mathbf{H}\mathbf{W}_{\text{down},h}$ denotes the projected input hidden states.

3.3 Post-Training LACE for Collaborative Search

The data pipeline defines what collaborative behavior should look like, and Lattice Attention provides the channel through which it can happen. The final step is to train the model so this channel becomes useful rather than passive. Our training framework consists of supervised fine-tuning and reinforcement learning.

Supervised Fine-Tuning. We initialize LACE from a pre-trained base model and fine-tune it on our curated multi-thread reasoning data [23]. To stabilize the newly inserted lattice attention pathway, the initial SFT steps update only the lattice attention parameters using the standard cross-entropy loss [25], adapting cross-thread communication while preserving the base model's causal reasoning ability. We then continue with full-parameter fine-tuning on the same data to teach the model the multi-thread reasoning format introduced in Section 3.1, including summaries and self-selection tags. We additionally apply random thread shuffling during training with a fixed probability threshold. This prevents the model from overfitting to fixed thread identities and encourages it to rely on cross-thread information rather than reasoning solely from intra-thread context.

Lattice GRPO. Finally, we optimize the model with Lattice GRPO, our multi-thread extension of GRPO [11, 28, 29]. Following baseline GRPO, the policy model rolls out G reasoning groups. In Lattice GRPO, each group contains T thread completions generated simultaneously, enabling reward computation over the full thread group. Along each thread, we prompt the model to generate a self-assessment enclosed in `<summary>...</summary>` and tag each thread with one of BEST, SUCCESS, and FAIL. This turns RL into a direct pressure on both exploration and communication, since the model must generate diverse candidates and identify which thread is most reliable.

Thread-Aggregated Reward. We aim for LACE to use cross-thread attention to explore diverse solution paths in search of a correct answer. Because the threads interact during generation, we assign credit based on their collective outcome. We combine an accuracy reward for correct solutions and reliable self-selection with a diversity reward for exploring different reasoning strategies:

$$r^{(g)} = R_{\text{acc}}(\mathbf{x}, \mathbf{y}^{(g)}) + \lambda_{\text{div}} R_{\text{div}}(\mathbf{y}^{(g)}). \tag{1}$$

Diversity reward. To encourage different solution strategies, we measure diversity using the average pairwise cosine distance between embeddings of the threads' reasoning paths [52]: $R_{\text{div}}(\mathbf{y}) = \frac{2}{T(T-1)} \sum_{i<j} \big(1 - \cos(\mathbf{e}^{(i)}, \mathbf{e}^{(j)})\big)$, where $\mathbf{e}^{(t)}$ is the embedding of thread t's reasoning path.
Accuracy reward. The accuracy reward encourages the group to produce correct solutions and identify a reliable answer among them. We define $R_{\text{acc}}^{\text{gated}} = R_{\text{corr}}$ if single BEST, else -1, where R_{corr} measures the correctness of the threads. The reward can also be softened by assigning separate credit to the self-selection format and answer correctness: $R_{\text{acc}}^{\text{soft}} = R_{\text{corr}} + \lambda_{\text{fmt}} R_{\text{format}}$, where R_{format} measures compliance with the self-selection format and λ_{fmt} controls its weight. This allows correct solutions to be credited even when the selection format is imperfect.

The resulting group advantage $A^{(g)}$ is shared across all T threads, i.e., $A^{(g,t)} = A^{(g)}$, so that each thread is optimized toward the group's collective outcome.

Lattice GRPO Objective. The final Lattice GRPO objective combines the clipped surrogate loss with a KL penalty:

$$\mathcal{L}_{\text{Lattice-GRPO}}(\theta) = -\mathbb{E}\left[\sum_{g=1}^{G}\sum_{t=1}^{T}\frac{1}{L^{(g,t)}}\sum_{l=1}^{L^{(g,t)}}\min\Big(\rho_l^{(g,t)}A^{(g)}, \text{clip}(\rho_l^{(g,t)}, 1-\epsilon, 1+\epsilon)A^{(g)}\Big)\right] + \beta D_{\text{KL}} \tag{2}$$

where $\rho_l^{(g,t)}$ is the importance sampling ratio for token l in thread t of group g, with the policy distribution conditioned on the thread's own prefix and peer same-step lattice states. $L^{(g,t)}$ is the completion length, and D_{KL} is the KL divergence from a reference policy.

4 Experiments

We conduct experiments to evaluate the effectiveness of our proposed LACE framework on challenging math and reasoning benchmarks. We compare our method against strong baselines, assessed by accuracy and exploration diversity in Section 4.1. We then analyze its **emergent cross-thread mechanisms** [37, 41] and ablate the data and training pipeline in Section 4.2, showing how collaborative exploration and self-regulated behaviors arise from the lattice architecture.

Implementation Details. We implement our method using Qwen3 [44] models with 1.7B and 4B parameters, as well as LLaMA-3.1 [9] with 8B parameters. The Qwen3-based LACE variants are denoted Lattice-1.7B and Lattice-4B. Training data for the Qwen3 and LLaMA-3.1 models is curated from DAPO [49] and DART-Math [34], respectively, using our data curation pipeline described in Section 3.1. Four threads are applied in the lattice structure for most experiments. Lattice attention layers are inserted every other layer, starting from the middle to the last original layer. Detailed thread-scaling computation costs are reported in Section B.4, showing negligible FLOPs overhead ($< 1.3\%$), modest memory overhead (roughly 1.6–15%), and latency overhead that is primarily memory-bandwidth bound. The detailed training hyperparameters and data statistics are provided in Section A.

4.1 Quantitative Results

Baselines and Metrics. We compare our LACE framework against two strong baselines: **Independent** and **Isolated Parallel**. The Independent baseline is a standard single-thread model trained with conventional SFT and RL; at inference time, we sample multiple independent trajectories and report both mean accuracy and majority voting. The Isolated Parallel baseline uses a training recipe aligned with LACE, but keeps the standard base model without inserted lattice layers or cross-thread attention, so the threads are trained and decoded in parallel yet remain independent. To ensure a fair comparison, all methods undergo SFT and RL stages; for LACE, the initial SFT steps are used to initialize the newly inserted lattice layers before standard full-parameter fine-tuning. Table 2 additionally compares

against Gen-Judge, Seq-Ref, ToT, GoT, and Hogwild! [27]. We report **Accuracy**, **Mean@4**, and **Pass@4**. As described in Section 3.1, LACE and Isolated Parallel are trained to self-select the optimal thread. For these multi-thread methods, accuracy is computed from the answer in the thread that the model marks as BEST among four simultaneous threads. For Independent, which lacks a self-selection mechanism, Accuracy uses majority voting. Mean@4 averages correctness across the four samples, while Pass@4 measures the fraction of problems with at least one correct sample [4]. Additional results with the gated accuracy reward $R_{\text{acc}}^{\text{gated}}$ and embedding-based diversity are reported in Section B.1.

Main Results. As summarized in Table 1, LACE achieves the highest accuracy after RL across all benchmarks and model scales. The gains over both Independent + Vote and Isolated Parallel show the benefit of cross-thread interaction under the same four-sample decoding budget. After RL, higher Mean@4 than Independent suggests effective **collaborative reasoning**, where information shared across threads helps improve individual solutions. Higher Pass@4 reflects broader candidate coverage, consistent with **diverse exploration** that increases the chance that at least one path reaches a correct solution. Lattice-1.7B also shows the benefit of **self-selection**: its selected-answer accuracy exceeds both Mean@4 and majority voting over the same four threads on all three benchmarks (Figure 4c). When examining failure cases, we find that LACE has a lower average BEST mislabeling rate than Isolated Parallel under both accuracy rewards, while its remaining errors on AIME mainly stem from limited candidate coverage (Section B.2). Complementary analysis with $R_{\text{acc}}^{\text{gated}}$ in Figure 4 shows an entropy rebound and increasing diversity during RL. This combination of high accuracy and meaningful diversity validates the framework's ability to foster collaborative reasoning and enhance the reliability of multi-threaded generation.

Table 1: **Comparisons with baselines on math and reasoning benchmarks.** For LACE and Isolated Parallel, Accuracy evaluates the answer self-labeled as BEST; Independent uses majority voting (†). Mean@4 reports average accuracy across four samples, while Pass@4 measures the fraction of problems with at least one correct sample. Scores are percentages, reported as mean ± standard deviation over at least three evaluation runs. The best and second-best results within each model and metric are bolded and underlined, respectively. All LLaMA-3.1-8B results are reported after SFT and RL. Minerva denotes Minerva-Math.

Model	Method	Train	AIME 25			AIME 24			LiveBench		
			Accuracy	Mean@4	Pass@4	Accuracy	Mean@4	Pass@4	Accuracy	Mean@4	Pass@4
Qwen3 1.7B	*Independent + Vote*	SFT	3.8†±2.1	4.7±1.4	12.9±6.0	6.7†±4.7	5.5±2.6	12.5±5.6	22.3†±0.9	33.5±1.3	64.4±1.3
		SFT → RL	13.8†±4.2	12.7±2.5	23.8±4.5	<u>18.3</u>†±4.7	<u>16.9</u>±3.1	<u>31.2</u>±5.9	37.5†±2.5	40.1±1.5	68.4±2.9
	Isolated Parallel	SFT	2.5±3.0	3.6±1.7	10.8±5.3	1.2±1.7	4.2±1.6	9.2±5.0	34.0±2.4	34.7±0.9	65.5±1.8
		SFT → RL	<u>16.2</u>±4.2	<u>14.6</u>±2.4	<u>27.9</u>±7.1	12.5±3.0	13.5±2.3	27.1±3.8	<u>48.1</u>±3.0	**43.3**±1.3	**72.8**±1.6
	LACE	SFT	3.3±3.1	3.8±1.1	10.8±2.4	4.2±3.9	4.5±1.7	9.6±4.5	36.5±3.2	35.8±1.4	66.3±2.2
		SFT → RL	**19.2**±5.0	**18.2**±1.6	**30.8**±4.3	**25.0**±5.0	**20.5**±2.4	**36.2**±4.5	**48.6**±1.3	<u>42.3</u>±0.7	<u>70.4</u>±1.8
Qwen3 4B	*Independent + Vote*	SFT	15.6†±3.8	12.2±0.5	24.4±5.1	12.2†±3.8	12.2±1.7	31.1±10.7	43.8†±2.6	50.4±2.3	79.6±1.9
		SFT → RL	<u>23.3</u>†±0.0	<u>23.9</u>±0.5	<u>33.3</u>±3.3	<u>31.1</u>†±3.8	<u>26.4</u>±1.7	<u>43.3</u>±3.3	<u>60.7</u>†±1.4	<u>63.3</u>±1.4	<u>85.0</u>±2.1
	Isolated Parallel	SFT	2.2±1.9	10.6±2.4	24.4±3.8	6.7±0.0	12.2±3.4	26.7±3.3	27.5±2.9	48.3±0.9	78.2±2.6
		SFT → RL	13.3±5.8	14.4±1.9	26.7±6.7	16.7±3.3	16.1±2.5	26.7±6.7	44.8±2.7	56.9±1.6	82.6±1.9
	LACE	SFT	7.8±7.7	12.8±3.4	26.7±0.0	5.6±1.9	13.6±1.3	27.8±3.8	25.4±3.2	50.9±0.4	80.8±1.4
		SFT → RL	**27.8**±1.9	**24.2**±2.2	**42.2**±6.9	**33.3**±0.0	**30.6**±0.5	**55.6**±5.1	**64.1**±3.5	**68.2**±1.4	**87.7**±1.4

Model	Method	MATH-Hard			OlympiadBench			Minerva			LiveBench		
		Accuracy	Mean@4	Pass@4	Accuracy	Mean@4	Pass@4	Accuracy	Mean@4	Pass@4	Accuracy	Mean@4	Pass@4
LLaMA-3.1 8B	*Independent + Vote*	<u>20.1</u>†±0.6	<u>19.4</u>±0.2	<u>36.0</u>±0.4	15.9†±1.0	<u>16.1</u>±0.6	<u>30.6</u>±1.0	24.0†±1.1	<u>23.7</u>±0.3	<u>40.6</u>±1.0	24.0†±1.6	<u>33.6</u>±1.5	<u>61.1</u>±0.5
	Isolated Parallel	18.7±0.3	18.8±0.3	34.9±0.7	<u>16.1</u>±1.3	15.5±0.7	29.7±1.2	<u>25.0</u>±0.8	23.6±0.9	38.8±1.1	<u>27.7</u>±1.8	21.6±0.6	50.4±1.6
	LACE	**21.5**±0.7	**21.8**±0.3	**40.9**±0.4	**19.3**±0.6	**18.7**±0.7	**34.7**±0.6	**26.2**±1.5	**27.1**±0.8	**45.4**±1.4	**28.7**±0.9	**34.8**±0.6	**63.5**±1.2

Additional Reasoning Baselines. Table 2 shows that LACE achieves the highest accuracy on both AIME benchmarks with Qwen3-4B, with lower reported latency and token cost than Gen-Judge and Seq-Ref. ToT uses fewer resources but achieves lower accuracy. With LLaMA-3.1-8B, LACE also achieves higher accuracy and lower token cost than Hogwild! across all four benchmarks.

Table 2: **Comparisons with additional reasoning baselines.** Gen-Judge independently samples four paths and uses the same model to select the best. Seq-Ref sequentially generates four paths, directly revising from all prior completed paths without a separate feedback step. Accuracy is mean $\pm$ standard deviation. Latency (seconds) and token cost (total generated tokens) are reported per question averagely. Best results are bolded; second-best Qwen3 results are underlined.

Model	Method	AIME 25			AIME 24		
		Acc.	Latency	Token Cost	Acc.	Latency	Token Cost
Qwen3 4B	*Gen-Judge* [55]	<u>25.6</u>$\pm$1.9	132	18,841	<u>26.7</u>$\pm$3.3	166	21,571
	Seq-Ref [22]	22.2$\pm$3.8	208	14,848	21.1$\pm$5.1	257	18,198
	ToT [48]	16.7$\pm$0.0	**47**	**10,922**	21.1$\pm$5.1	**49**	**11,071**
	GoT [3]	18.9$\pm$6.9	76	<u>12,141</u>	20.0$\pm$5.8	73	12,183
	LACE	**27.8**$\pm$1.9	<u>68</u>	12,513	**33.3**$\pm$0.0	<u>59</u>	<u>11,629</u>

Model	Method	MATH-Hard			OlympiadBench			Minerva			LiveBench		
		Acc.	Latency	Token Cost	Acc.	Latency	Token Cost	Acc.	Latency	Token Cost	Acc.	Latency	Token Cost
LLaMA-3.1 8B-Instruct	*Hogwild!* [27]	13.0$\pm$0.4	218	9,282	13.2$\pm$1.0	228	9,691	16.8$\pm$0.8	147	6,457	26.9$\pm$1.7	275	11,295
	LACE	**21.5**$\pm$0.7	**39**	**6,221**	**19.3**$\pm$0.6	**42**	**6,565**	**26.2**$\pm$1.5	**29**	**4,487**	**28.7**$\pm$0.9	**37**	**5,824**

Agentic Benchmarks. Beyond math benchmarks, we additionally train and evaluate Lattice-1.7B on HotpotQA [46], TextWorldCookAgent from TALES [7, 6], and WebShop [47]. In our setup, HotpotQA evaluates answer correctness after multi-step retrieval, while TextWorldCook and WebShop evaluate task completion through environment scores and success rates. Table 3 shows that LACE achieves the highest Accuracy, Mean@4, and Pass@4 on HotpotQA and the highest Score and Success rate on TextWorldCook and WebShop among the matched baselines.

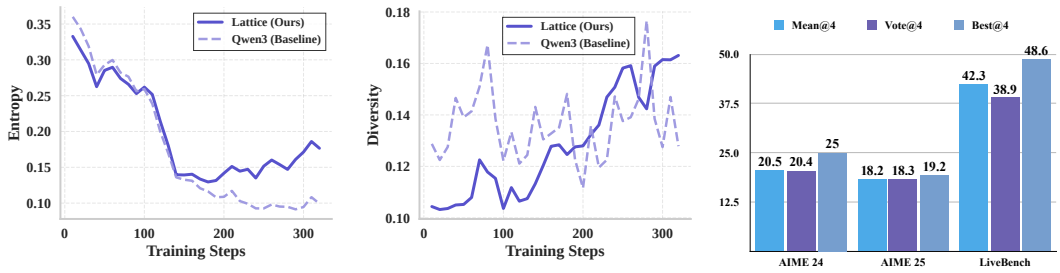
Table 3: **Agentic benchmarks with Qwen3-1.7B.** HotpotQA reports exact match; † denotes majority voting for Independent. TextWorldCook and WebShop report best-of-four scores and success rates. Results are mean $\pm$ std.

Model	Method	HotpotQA			TextWorldCook		WebShop	
		Accuracy	Mean@4	Pass@4	Score	Success rate	Score	Success rate
Qwen3 1.7B	*Independent*	25.5$^{\dagger}\pm$0.9	25.8$\pm$0.5	32.0$\pm$0.2	<u>66.5</u>$\pm$0.6	<u>41.3</u>$\pm$0.6	53.6$\pm$0.2	14.7$\pm$0.2
	Isolated Parallel	<u>26.8</u>$\pm$0.8	<u>27.6</u>$\pm$0.6	<u>34.3</u>$\pm$0.4	63.0$\pm$0.7	38.0$\pm$1.0	<u>56.2</u>$\pm$0.4	<u>18.6</u>$\pm$0.4
	LACE	**28.6**$\pm$0.5	**28.3**$\pm$0.5	**34.4**$\pm$0.7	**67.2**$\pm$0.6	**44.3**$\pm$2.1	**57.2**$\pm$0.4	**21.6**$\pm$0.7

4.2 Analysis and Ablations

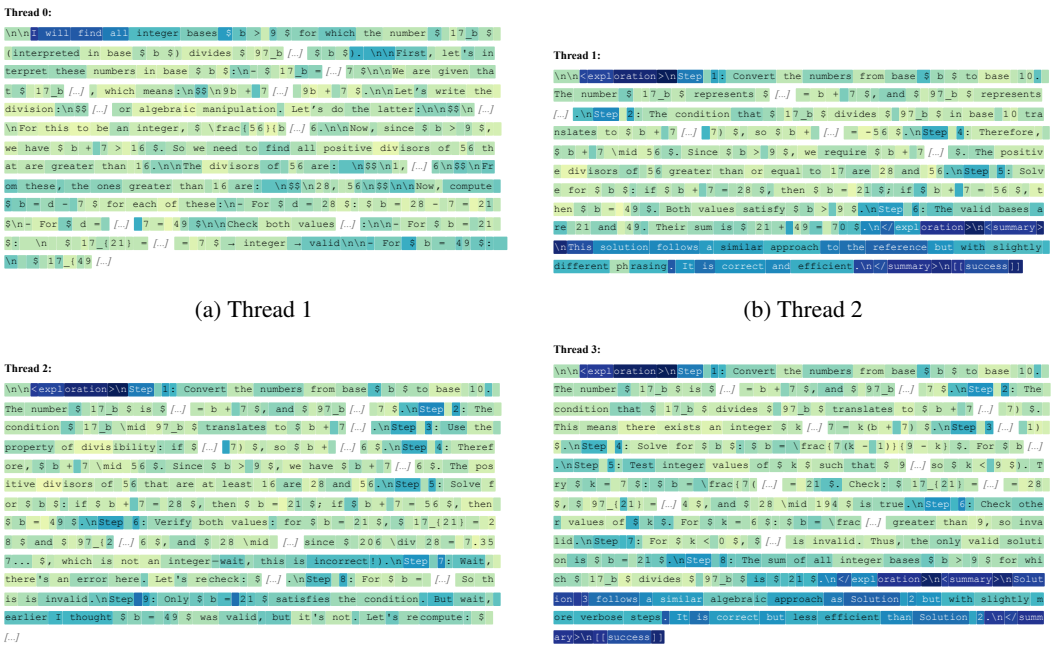
LACE avoids mode collapse and maintains diverse exploration. We investigate the training dynamics to understand how the model learns to explore. As shown in Figure 4, the baseline model suffers from mode collapse, evidenced by a continuous drop in entropy. In contrast, LACE exhibits a unique pattern where entropy slowly rebounds after an initial decrease, accompanied by a steady rise in diversity scores. This suggests that the synergy between lattice attention and diversity rewards successfully enables **learned exploration**, allowing the model to actively seek diverse reasoning paths rather than converging to a single mode. Section B.1 and Table 9 further show that the diversity reward behaves like a training-time temperature: the default weight gives the highest self-selection accuracy, while a larger weight keeps Pass@4 at a similar level but makes stable self-selection harder.

LACE communicates most when threads need peer context. Lattice Attention utilizes a gating mechanism to regulate cross-thread information flow, providing a unique lens to analyze collaborative intensity across both temporal and semantic dimensions. Temporally, Figure 6 illustrates that



(a) Entropy (b) Diversity (c) Self-selection vs. voting.

Figure 4: **LACE improves self-selection by fostering more effective exploration.** (**a**) and (**b**) compare the entropy and diversity dynamics of Lattice-4B against the Qwen3-4B baseline during RL training under $R_{\text{acc}}^{\text{gated}}$: while the baseline suffers from entropy collapse after roughly 150 steps, Lattice-4B exhibits a stable entropy rebound and an overall upward trend in diversity. (**c**) compares Mean@4, Vote@4, and self-selected Best@4 predictions for Lattice-1.7B. This demonstrates that the model can select a single reliable candidate rather than merely generate multiple answers.



(a) Thread 1 (b) Thread 2

(c) Thread 3 (d) Thread 4

Figure 5: **LACE coordinates information sharing across all four threads at critical reasoning steps.** We visualize cross-thread intensity (gate score) on an AIME 25 example from Lattice-1.7B, where darker shading indicates stronger cross-thread interaction. High cross-thread communication in latent space consistently emerges during exploration, self-assessment, and evaluation, and the generated summaries explicitly compare peer solutions.

information flow peaks during the core reasoning phase, exhibiting a diverge-then-converge strategy coinciding with the classical framework of human intelligence proposed by Guilford [10], which distinguishes between divergent production and convergent production to arrive at a final correct solution. Semantically, Figure 5 reveals that this interaction is content-specific across all four threads: high gate scores align with tokens that require peer information, including exploration markers, summary steps, and self-assessment tags. The generated text provides complementary evidence: for example, Thread 4 explicitly compares its path against other threads in its summary, indicating that the model uses peer trajectories when evaluating candidate solutions. This suggests that LACE enables **need-aware collaboration**: the model increases cross-thread communication when comparing, summarizing, or selecting among candidate paths, rather than exchanging information uniformly

Table 4: **Diversity-aware data curation improves accuracy and diversity.** We compare training recipes under the 1.7B ablation protocol with 200 RL steps and $R_{\text{acc}}^{\text{gated}}$.

Training Recipe	AIME 25			LiveBench		
	Accuracy	Div.	Format	Accuracy	Div.	Format
RL only (w/o SFT)	0.0	0.285	26.7	2.1	0.244	31.5
SFT → RL w/ Parallel-R1	0.0	0.203	50.0	4.1	0.148	20.5
SFT → RL w/ Our Data	**13.3**	**0.608**	**73.3**	**14.5**	**0.276**	**36.5**

Table 5: **Our curated training data induces both more diverse reasoning paths and stronger cross-thread communication than Parallel-R1 data after SFT.** We compare our curated data with Parallel-R1 data using Edit Distance, Embedding Dissimilarity, and Cross-Thread Attention Ratio. Higher values indicate greater diversity or a higher relative weight on peer-thread attention.

Data	Diversity ↑		Cross-Thread Attention ↑
	Edit Distance	Embedding Dissimilarity	
Parallel-R1 Data + *SFT*	0.13	0.00	0.307
Our Data + *SFT*	**0.77**	**0.12**	**0.353**

throughout generation. Sections C.1 and C.2 further show that synchronization concentrates in middle-to-late layers especially near Layer 26 across both model scales.

Diversity-aware data and SFT improve collaboration and self-selection. Table 5 shows that our curated data produces more diverse threads and stronger cross-thread attention than Parallel-R1 data. Table 4 shows that SFT improves self-selection under $R_{\text{acc}}^{\text{gated}}$. However, this reward's strict format requirement leaves open whether SFT mainly teaches the output format or the ability to select a correct path. We therefore further ablate SFT under the format-relaxed reward $R_{\text{acc}}^{\text{soft}}$ on LLaMA-3.1-8B (Section B.3). RL alone achieves higher Mean@8 but lower Accuracy than SFT→RL, suggesting that SFT supports reliable self-selection even when the strict format gate is relaxed. To measure the intensity of cross-thread communication, we define the cross-thread ratio as $r_{\text{cross}} = \frac{\bar{A}_{\text{cross}}}{\bar{A}_{\text{self}}+\bar{A}_{\text{cross}}}$, where $\bar{A}_{\text{self}}$ denotes the average attention weight assigned to keys from the same thread and $\bar{A}_{\text{cross}}$ denotes the average attention weight assigned to keys from peer threads. As shown in Table 5, our diversity-aware data pipeline yields substantially higher thread diversity and cross-thread attention ratios compared to naïve parallel sampling, validating that data quality is the prerequisite for emergent collaboration.

5 Conclusion

Current reasoning methods mainly scale the depth of individual chains, while leaving reasoning paths largely isolated. To address this, we introduce LACE, which adds a lightweight cross-thread side path to transformers and enables **implicit token-level cross-thread interaction during generation**, rather than post-hoc aggregation or isolated parallel sampling. With synthetic collateral-thinking data and lattice-based RL, LACE turns redundant samples into interactive exploration and delivers consistent gains over matched baselines. Overall, our findings suggest that stronger reasoning may come not only from stronger individual threads, but also from emergent cross-thread collaboration.

6 Limitations

Although lattice attention adds less than 1.3% FLOPs, cross-thread communication still incurs latency overhead (Tables 11 and 12). We observe that RL on mathematical data alone can destabilize output formatting on out-of-domain benchmarks such as LiveBench [40]. Selecting BEST threads during data generation also requires criteria tailored to each domain, and extending the model to open-domain agent tasks such as WebArena [58] requires further adaptation. Following Gated Attention [24], we insert lattice attention after the token-axis attention output; other placements, such as after the feed-forward block, are not ablated in this work. The number of threads is also fixed at inference and does not adapt to problem difficulty.

Acknowledgements and Disclosures

This work is supported by Amazon's Gift Fund.

References

[1] Chenyang An, Zhizhen Qin, Soonho Kong, and Cedric Flamant. DeRL: Diverse-exploration reinforcement learning for large language models improves mathematical reasoning. Under review at ICLR 2025, 2024. URL https://openreview.net/forum?id=ZIYYeTkZQ4.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[3] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of Thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, 2024. doi: 10.1609/aaai.v38i16.29720. URL https://ojs.aaai.org/index.php/AAAI/article/view/29720.

[4] Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] 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.

[6] Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532, 2018.

[7] Christopher Cui, Xingdi Yuan, Ziang Xiao, Prithviraj Ammanabrolu, and Marc-Alexandre Côté. Tales: Text-adventure learning environment suite. *arXiv preprint arXiv:2504.14128*, 2025. URL https://arxiv.org/abs/2504.14128.

[8] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

[9] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

[10] Joy Paul Guilford. *The nature of human intelligence.* McGraw-Hill, 1967.

[11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[12] Lu Hong and Scott E Page. Groups of diverse problem solvers can outperform groups of high-ability problem solvers. *Proceedings of the National Academy of Sciences*, 101(46): 16385–16389, 2004.

[13] Chan-Jan Hsu, Davide Buffelli, Jamie McGowan, Feng-Ting Liao, Yi-Chang Chen, Sattar Vakili, and Da-shan Shiu. Group think: Multiple concurrent reasoning agents collaborating at token level granularity. *arXiv preprint arXiv:2505.11107*, 2025.

[14] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

[15] Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Kaito Ariu, Kenshi Abe, Mitsuki Sakamoto, and Eiji Uchibe. Evaluation of best-of-n sampling strategies for language model alignment. *arXiv preprint arXiv:2502.12668*, 2025.

[16] Philip N Johnson-Laird. Mental models and human reasoning. *Proceedings of the National Academy of Sciences*, 107(43):18243–18250, 2010.

[17] Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

[18] Kenneth Kay, Jason E Chung, Marielena Sosa, Jonathan S Schor, Mattias P Karlsson, Margaret C Larkin, Daniel F Liu, and Loren M Frank. Constant sub-second cycling between representations of possible futures in the hippocampus. *Cell*, 180(3):552–567, 2020.

[19] Elliot Kim, Avi Garg, Kenny Peng, and Nikhil Garg. Correlated errors in large language models. *arXiv preprint arXiv:2506.07962*, 2025.

[20] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[21] Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, et al. From system 1 to system 2: A survey of reasoning large language models. *arXiv preprint arXiv:2502.17419*, 2025.

[22] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594, 2023.

[23] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[24] Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.

[25] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[26] Amin Rakhsha, Kanika Madan, Tianyu Zhang, Amir-massoud Farahmand, and Amir Khasahmadi. Majority of the bests: Improving best-of-n via bootstrapping. *arXiv preprint arXiv:2511.18630*, 2025.

[27] Gleb Rodionov, Roman Garipov, Alina Shutova, George Yakushev, Erik Schultheis, Vage Egiazarian, Anton Sinitsin, Denis Kuznedelev, and Dan Alistarh. Hogwild! inference: Parallel LLM generation via concurrent attention. *arXiv preprint arXiv:2504.06261*, 2025.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[29] 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.

[30] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

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

[32] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

[33] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[34] Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. DART-Math: Difficulty-aware rejection tuning for mathematical problem-solving. *arXiv preprint arXiv:2407.13690*, 2024. URL https://arxiv.org/abs/2407.13690.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[36] 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.

[37] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[38] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[39] 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.

[40] Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. Livebench: A challenging, contamination-free LLM benchmark. In *The Thirteenth International Conference on Learning Representations*, 2025.

[41] Anita Williams Woolley, Christopher F Chabris, Alex Pentland, Nada Hashmi, and Thomas W Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004):686–688, 2010.

[42] Tong Wu, Yang Liu, Jun Bai, Zixia Jia, Shuyi Zhang, Ziyong Lin, Yanting Wang, Song-Chun Zhu, and Zilong Zheng. Native parallel reasoner: Reasoning in parallelism via self-distilled reinforcement learning. *arXiv preprint arXiv:2512.07461*, 2025.

[43] 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. *arXiv preprint arXiv:2408.00724*, 2024.

[44] 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.

[45] Sohee Yang, Elena Gribovskaya, Nora Kassner, Mor Geva, and Sebastian Riedel. Do large language models latently perform multi-hop reasoning? *arXiv preprint arXiv:2402.16837*, 2024.

[46] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380. Association for Computational Linguistics, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

[47] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757, 2022. URL https://arxiv.org/abs/2207.01206.

[48] 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.

[49] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[50] Yijiong Yu, Wei Wang, Ran Chen, and Ji Pei. Accelerate parallelizable reasoning via parallel decoding within one sequence. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 9018–9025, 2025.

[51] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023.

[52] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

[53] Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2024, 2024.

[54] Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

[55] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

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

[57] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

[58] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023. URL `https://arxiv.org/abs/2307.13854`.

A Implementation Details

A.1 Model Initialization and Training

We implement LACE using Qwen3 models with 1.7B and 4B parameters, as well as LLaMA-3.1 with 8B parameters. The 1.7B model is initialized from Qwen3-1.7B with disabled thinking for efficient training, the 4B model from Qwen3-4B-Instruct-2507, and the 8B model from LLaMA-3.1-8B-Instruct. All three models undergo SFT and RL training as described in Section 3.3. During the initial SFT steps, we freeze the backbone and update only the lattice attention parameters before unfreezing the full model for the remaining SFT steps. Training data for Qwen3 and LLaMA-3.1 is curated from DAPO and DART-Math, respectively, using the pipeline in Section 3.1.

For the Qwen3 models, the SFT learning rate is 1×10^{-5} with a cosine scheduler, and the RL learning rate is 1×10^{-6}. We use the AdamW optimizer with a weight decay of 0.01 and betas of (0.9, 0.95).

A.2 Reward and Evaluation Settings

For LACE and Isolated Parallel, we use $R_{\text{acc}}^{\text{soft}}$ for the experiments in Table 1. Experiments with $R_{\text{acc}}^{\text{soft}}$ use TRL 1.9.0 with an SGLang rollout engine, while those with $R_{\text{acc}}^{\text{gated}}$ use TRL 0.25.1 with a Hugging Face rollout engine. The $R_{\text{acc}}^{\text{soft}}$ setting uses $2\times$ the GPU count and $2\times$ the maximum rollout generation length of the $R_{\text{acc}}^{\text{gated}}$ setting.

For Table 1, we evaluate the 1.7B, 4B, and 8B models after 250, 400, and 150 RL steps, respectively; the training seed is fixed to 42 for all three model scales. For this setting, we compute the conditional probability of the BEST label for each thread under teacher forcing and select the answer from the thread with the highest probability. For the Qwen3 experiments with $R_{\text{acc}}^{\text{gated}}$, the 1.7B model is evaluated after 200 RL steps on AIME 24 and AIME 25 and after 350 RL steps on LiveBench, while the 4B model is evaluated after 320 RL steps on all three benchmarks.

Agentic benchmarks. For Table 3, HotpotQA uses a fixed 500-question subset of the 7,405-example `fullwiki` validation split and four decoding seeds. TextWorldCook uses the 100 TALES CookingWorld test games with repeated evaluation runs. WebShop uses all 500 instructions from the upstream test split and four decoding seeds.

For HotpotQA, we use rewards based on $R_{\text{acc}}^{\text{soft}}$ with exact-match (EM) scoring. LACE and Isolated Parallel use prompts with `[[best]]` labels.

For TextWorldCook and WebShop, we directly use the scores returned by the environments as rewards. LACE and Isolated Parallel use the maximum score across four threads, while Independent uses its own thread's score. Prompts for these tasks omit `[[best]]` labels.

A.3 Training Data Statistics

Table 6: **Training and evaluation data statistics used in this work.**

Training Set	Stage	Questions
Lattice-1.7B	SFT	2,409
Lattice-1.7B	RL	1,959
Lattice-4B	SFT	2,409
Lattice-4B	RL	6,474
Lattice-LLaMA-8B	Midtrain + SFT	3,291
Lattice-LLaMA-8B	RL	11,365
Lattice-1.7B-Subset	SFT	576
Lattice-1.7B-Subset	RL	960
DAPO-Test100	Held-out Test	100

All SFT splits contain at least four candidate chains-of-thought per problem. We use the subset split only for the additional scaling and ablation experiments in Section B.

B Additional Experimental Results

B.1 Results with the Gated Accuracy Reward ($R_{\text{acc}}^{\text{gated}}$)

Table 7 reports additional Qwen3 comparisons using the gated accuracy reward $R_{\text{acc}}^{\text{gated}}$ and training settings described in Section A, including embedding-based diversity and majority-voting results.

Table 7: **Comparisons with baselines on math and reasoning benchmarks. Independent** is a single-thread SFT+RL baseline, while **Isolated Parallel** uses the same multi-thread training recipe as LACE without lattice layers or cross-thread attention. We report **Accuracy** and **Diversity** (Div.). By default, Independent reports Mean@4, while Isolated Parallel and LACE use the answer from the thread selected by the model as BEST among four simultaneous threads; +Vote denotes replacing the default selection rule with majority voting.

Model	Method	Train	AIME 25		AIME 24		LiveBench	
			Accuracy	Div. ↑	Accuracy	Div. ↑	Accuracy	Div. ↑
Qwen3 1.7B	*Independent*	SFT → RL	8.1 ±0.5	0.07	9.7 ±1.0	0.07	31.7 ±1.4	0.06
	+ *Vote*	SFT → RL	7.8 ±1.9	0.07	11.1 ±1.9	0.07	30.0 ±1.0	0.06
	Isolated Parallel	SFT	1.1 ±1.9	0.10	2.2 ±1.9	0.11	17.2 ±2.6	**0.11**
		SFT → RL	6.7 ±3.3	0.16	4.4 ±1.9	0.19	28.9 ±2.9	0.10
	+ *Vote*	SFT → RL	2.2 ±1.9	0.16	2.2 ±1.9	0.19	23.8 ±0.6	0.10
	LACE	SFT	4.4 ±5.1	0.10	1.1 ±1.9	0.11	25.3 ±0.5	0.11
		SFT → RL	**13.3** ±3.3	**0.28**	**14.4** ±3.8	**0.24**	**33.6** ±3.1	0.11
Qwen3 4B	*Independent*	SFT → RL	11.7 ±2.2	0.06	13.6 ±1.9	0.07	**50.0** ±0.7	0.15
	+ *Vote*	SFT → RL	11.1 ±3.8	0.06	15.6 ±3.8	0.07	43.9 ±1.6	0.15
	Isolated Parallel	SFT	10.0 ±0.0	0.09	11.7 ±2.4	0.10	29.9 ±2.0	0.11
		SFT → RL	11.1 ±5.1	0.10	12.2 ±1.9	0.10	43.6 ±1.4	0.15
	+ *Vote*	SFT → RL	6.7 ±3.3	0.10	13.3 ±3.3	0.10	45.9 ±1.1	0.15
	LACE	SFT	10.0 ±0.0	0.09	7.8 ±1.9	0.09	21.3 ±2.5	0.10
		SFT → RL	**16.7** ±3.3	**0.16**	**18.9** ±1.9	**0.16**	47.9 ±1.4	**0.16**

Additional Baseline Comparisons. As shown in Table 8, judge-based post-hoc selection [55] is more accurate with a stronger external judge in this evaluation setting, but it requires additional decoding and verification cost. Sequential refinement [22] similarly incurs substantial extra decoding length. In contrast, LACE performs self-selection in situ during generation, retaining lower end-to-end latency and token usage while remaining competitive in final accuracy.

Table 8: **Additional selection-based baselines on AIME 24 and AIME 25.** We compare Lattice-4B against judge-based selection [55] and sequential refinement [22] baselines.

	AIME 24			AIME 25		
Method	**Acc**	**Latency/q**	**Tokens/q**	**Acc**	**Latency/q**	**Tokens/q**
LACE	20.0	61.2s	7,120	20.0	53.0s	7,012
Gen-Judge (self)	0.0	97.8s	10,573	0.0	105.3s	10,886
Gen-Judge (oracle-4o / 4.1)	20.0 / 16.7	86.7s / 98.6s	9,069 / 10,041	13.3 / 20.0	94.5s / 106.8s	9,499 / 10,237
Seq-Ref	16.7	$\sim$ 245s	15,446	13.3	$\sim$ 296s	14,369

Diversity Reward Sensitivity. We conduct this study on the subset training split using the 1.7B model. A useful intuition is that the diversity reward acts like a training-time temperature, reshaping the model's path distribution during training. As shown in Table 9, the default weight (1.0) gives the highest mean Acc on both datasets, but coverage does not follow the same order: Pass@4 stays within 2 points across the three weights, and on DAPO-Test100 it is highest without the diversity reward. A larger diversity weight (2.0) therefore keeps coverage at a similar level while making the correct path harder to self-select. Overall, the default weight is the best choice for self-selection accuracy rather than for coverage. The Acc differences are within about one standard deviation across evaluation runs, so we read this ordering as a trend.

Table 9: **Relative roles of accuracy reward and diversity reward.** Due to limited computational resources, we report results on the subset training split using Lattice-1.7B. The three models start from the same RL checkpoint and are trained for another 120 steps that differ only in the diversity weight. We report mean ± std over 8 evaluation runs. All numbers are percentages.

	Acc			Pass@4		
Dataset	**Div=0**	**Div=1.0**	**Div=2.0**	**Div=0**	**Div=1.0**	**Div=2.0**
AIME 24	5.8 ±2.4	**9.2** ±3.9	5.8 ±3.5	17.5 ±3.5	**18.8** ±5.0	18.3 ±5.0
DAPO-Test100	20.1 ±3.1	**21.6** ±3.4	20.8 ±2.7	**54.2** ±1.8	52.2 ±3.5	52.2 ±3.6

B.2 Failure Analysis

Coverage limits LACE on AIME. We decompose the remaining score deficit ($100 - \mathrm{Accuracy}$) into a coverage gap ($100 - \mathrm{Pass@4}$) and a selection gap ($\mathrm{Pass@4} - \mathrm{Accuracy}$). Coverage accounts for 89–96% of LACE's remaining error on AIME under $R_{\mathrm{acc}}^{\mathrm{gated}}$ (Section B.1) and 67–86% under $R_{\mathrm{acc}}^{\mathrm{soft}}$ (Table 1). Thus, most remaining AIME errors occur because none of the threads finds a correct solution. On LiveBench, selection plays a larger role, accounting for 42–66% of the score deficit across both reward settings.

LACE selects available solutions more reliably. To assess selection separately from coverage, we measure BEST mislabeling on questions where at least one thread earns credit and at least one thread emits BEST. A mislabel occurs when the selected thread scores below the best available candidate. Under $R_{\mathrm{acc}}^{\mathrm{gated}}$, Isolated Parallel's average mislabel rate is 52.9%, compared with 30.8% for LACE (Qwen3-4B on AIME 24 and AIME 25). Under $R_{\mathrm{acc}}^{\mathrm{soft}}$, the corresponding rates are 39.5% and 26.5%, averaged over both Qwen3 scales and all three benchmarks. On AIME, Isolated Parallel's labels show no statistically detectable advantage over random selection: its Accuracy stays within 1.6 points of Mean@4, while Pass@4 is 10–15 points higher. These results highlight LACE's advantage in selecting from its available candidates.

B.3 Role of the Synthetic SFT Data

To examine whether synthetic SFT mainly provides a cold start for the multi-thread output format or also affects exploration and self-selection, we train LACE on LLaMA-3.1-8B with RL alone and compare it with SFT→RL (Table 10). Both runs use $R_{\mathrm{acc}}^{\mathrm{soft}}$. The two settings achieve similar Pass@8 on both benchmarks. Mean@8 is similar on MATH-Hard and higher for RL alone on Minerva-Math. SFT→RL achieves higher mean Accuracy, with the clearest gain on MATH-Hard. Its Accuracy also exceeds its own Mean@8 on both benchmarks, whereas RL alone shows the opposite pattern. These results suggest that synthetic SFT helps the model turn available candidate solutions into a reliable final answer. In this setting, its benefit is more apparent in self-selection than in candidate coverage.

Table 10: **RL only vs. SFT→RL for LACE on LLaMA-3.1-8B.** This ablation uses a separate setting from Table 1. RL only starts from LLaMA-3.1-8B-Instruct; SFT→RL starts from an SFT checkpoint trained with a simplified version of our data pipeline on the first 3,000 DART-Math questions. Both runs use $R_{\mathrm{acc}}^{\mathrm{soft}}$ and are evaluated at step 150. We evaluate on MATH-Hard (1,324 questions) and Minerva-Math (272 questions), using T=8 threads, 4,096 tokens per thread, SGLang decoding with forced answer completion at the token limit, and dual grading. Results are mean ± standard deviation over four evaluation runs.

	MATH-Hard			Minerva-Math		
Setting	**Accuracy**	**Mean@8**	**Pass@8**	**Accuracy**	**Mean@8**	**Pass@8**
RL only	15.7 ±0.8	**18.0** ±0.2	**43.8** ±0.8	20.2 ±0.5	**21.9** ±0.3	**44.7** ±2.3
SFT → RL	**20.3** ±1.1	17.9 ±0.2	42.8 ±0.7	**22.1** ±1.3	19.3 ±0.8	44.3 ±1.8

B.4 Computational Overhead

We benchmark thread scaling on a single RTX PRO 6000 Blackwell 97GB GPU using bfloat16 precision, generation length 100, and context length 50. We compare vanilla independent sampling against LACE-enabled decoding across thread counts from $N = 1$ to $N = 128$. The results in

Table 11: **Thread-scaling micro-benchmark on Lattice-1.7B.** Memory is reported in MB. Red shading marks overhead increases.

	Vanilla					Lattice					Overhead			
N	**ms/step**	**TPS**	**TTFT (ms)**	**Mem**	**FLOPs**	**ms/step**	**TPS**	**TTFT (ms)**	**Mem**	**FLOPs**	**ms/step**	**TTFT**	**Mem**	**FLOPs**
1	20.8	48.1	18.1	3,911	1.90×10^{11}	–	–	–	–	–	–	–	–	–
4	20.5	195.4	18.7	4,003	7.60×10^{11}	28.4	140.8	25.8	4,069	7.69×10^{11}	+38.5%	+38.0%	+1.6%	+1.22%
8	20.5	391.2	18.6	7,993	1.52×10^{12}	28.7	279.1	26.0	8,128	1.54×10^{12}	+40.0%	**39.8%**	+1.7%	+1.22%
16	20.4	783.6	25.8	8,217	3.04×10^{12}	31.0	515.5	29.8	8,416	3.08×10^{12}	+51.9%	+15.5%	+2.4%	+1.22%
32	21.6	1,480	33.8	8,673	6.08×10^{12}	33.2	964.8	38.2	8,997	6.15×10^{12}	+53.7%	+13.0%	+3.7%	+1.23%
64	22.5	2,845	51.9	9,585	1.22×10^{13}	35.2	1,817	57.0	10,161	1.23×10^{13}	+56.4%	+9.8%	+6.0%	+1.25%
128	25.8	4,968	91.5	11,181	2.43×10^{13}	41.0	3,124	100.1	12,278	2.46×10^{13}	**58.9%**	+9.4%	**9.8%**	**1.29%**

Table 12: **Thread-scaling micro-benchmark on Lattice-4B.** Memory is reported in MB. Red shading marks overhead increases.

	Vanilla					Lattice					Overhead			
N	**ms/step**	**TPS**	**TTFT (ms)**	**Mem**	**FLOPs**	**ms/step**	**TPS**	**TTFT (ms)**	**Mem**	**FLOPs**	**ms/step**	**TTFT**	**Mem**	**FLOPs**
1	26.2	38.5	23.1	7,767	4.42×10^{11}	–	–	–	–	–	–	–	–	–
4	27.6	144.8	23.9	7,874	1.77×10^{12}	36.2	110.6	33.1	8,717	1.77×10^{12}	+31.2%	**38.5%**	+10.7%	+0.35%
8	26.3	304.6	27.3	15,745	3.54×10^{12}	36.2	221.2	33.5	17,417	3.55×10^{12}	+37.6%	+22.7%	+10.6%	+0.36%
16	26.6	602.7	36.4	16,007	7.07×10^{12}	39.6	403.6	41.8	17,766	7.10×10^{12}	+48.9%	+14.8%	+11.0%	+0.36%
32	28.8	1,112	61.5	16,551	1.41×10^{13}	42.1	759.7	66.6	18,473	1.42×10^{13}	+46.2%	+8.3%	+11.6%	+0.36%
64	29.6	2,161	108.5	17,641	2.83×10^{13}	45.4	1,409	115.5	19,887	2.84×10^{13}	+53.4%	+6.5%	+12.7%	+0.36%
128	33.3	3,842	201.4	19,594	5.66×10^{13}	65.4	1,958	211.4	22,506	5.68×10^{13}	**96.4%**	+5.0%	**14.9%**	**0.37%**

Tables 11 and 12 show that LACE adds negligible FLOPs overhead (below 1.3%), while the dominant cost is step latency, indicating that the extra overhead is primarily memory-bandwidth bound rather than compute bound. Memory overhead remains modest, and TTFT overhead decreases at larger thread counts. Notably, even 128 threads fit on a single GPU, requiring only about 12.3GB for Lattice-1.7B and 22.5GB for Lattice-4B.

B.5 Thread Scaling

Table 13: **Inference-time thread scaling on TextWorldCookAgent.** A model trained with four threads is evaluated at $k = 4, 8, 16$ without retraining. Mean metrics average over threads; Win Rate counts any-thread success. Results are mean $\pm$ std.

Metric	**4T**	**8T**	**16T**
Mean Score	38.6 $\pm$0.7	39.9 $\pm$0.8	**40.0** $\pm$0.7
Best Score	67.2 $\pm$0.6	74.7 $\pm$1.1	84.1 $\pm$1.2
Win Rate	44.3 $\pm$2.1	55.2 $\pm$1.3	69.6 $\pm$2.5
Mean Win Rate	17.8 $\pm$1.8	**18.5** $\pm$0.9	18.3 $\pm$0.6

Table 14: **Training-time thread scaling on LiveBench.** We compare 4-thread and 8-thread Lattice-1.7B models trained on the subset split.

Metric	**4T train**	**8T train**
Accuracy	7.0	20.5
Latency / query (s)	59.3	21.2
Tokens / query	5,967	2,107

The model maintains comparable per-thread performance at eight and sixteen decoding threads, with modest gains over four-thread decoding (Table 13); best-of-k scores increase with the candidate count. Training with eight threads improves Accuracy and decoding efficiency in the LiveBench comparison (Table 14).

C Cross-Thread Collaboration Analysis

C.1 Gate Score Statistics

In Section 4.2, we show an example thread and visualize the gate scores assigned to each token, concluding that higher gate scores correspond to more critical reasoning steps. Here, we provide additional statistics to support this observation. As shown in Figure 7, we compute the average gate scores of Lattice-1.7B across all threads and samples in AIME 25, and rank the top-20 tokens by their mean gate scores. The results reveal a clear pattern: tokens associated with **self-assessment and cross-thread evaluation** dominate the rankings. Notably, subword tokens of "exploration" (`expl`, `oration`) and evaluation-related tokens (`summary`, `evaluate`, `Solution`) rank highly, indicating that the model increases cross-thread attention during the exploration phase. Furthermore, self-selection tags (SUCCESS, BEST) also exhibit elevated gate scores, substantiating that the model actively references peer threads when making final judgments. This quantitative evidence supports our qualitative finding that LACE learns to selectively engage cross-thread communication at semantically meaningful positions, particularly during exploration and self-assessment stages.

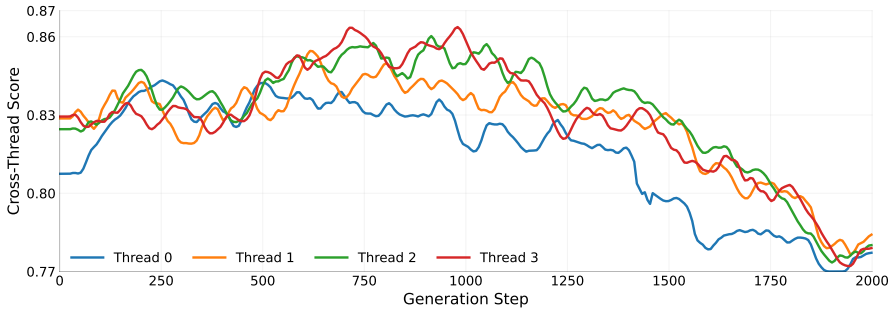


Figure 6: **Gate-score dynamics over decoding steps.** Information flow in the lattice peaks during the core reasoning phase before tapering off near the final answer, consistent with a diverge-then-converge pattern of collaborative reasoning.

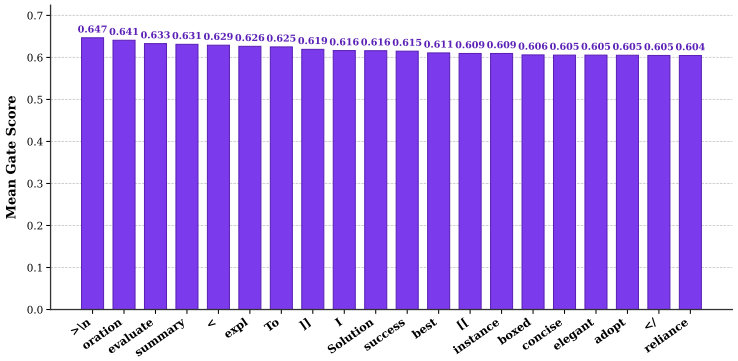


Figure 7: Average gate scores of top-20 tokens ranked by mean gate scores across all threads and samples in AIME 25 by Lattice-1.7B. Tokens related to self-assessment and cross-thread evaluation dominate the rankings, indicating that the model increases cross-thread attention during critical reasoning steps.

C.2 Per-Layer Synchronization

To further characterize where cross-thread collaboration is concentrated, we measure layer-wise synchronization under teacher forcing on AIME 25. We define synchronization as the product of the gate value and the cross-thread attention ratio, i.e., Sync $=$ Gate $\times\, r_{\text{cross}}$, which captures effective cross-thread information flow. Table 15 shows that synchronization is concentrated in middle-to-late layers, consistently serving as the stronger synchronization point across both model scales. RL also strengthens synchronization across nearly all layers, though the largest gains appear at different depths for 1.7B and 4B, suggesting scale-dependent collaboration patterns.

Table 15: **Per-layer synchronization analysis on AIME 25.** Sync denotes effective cross-thread information flow, computed as gate × cross-thread ratio. Green row shading highlights RL synchronization profiles, and green cells in Δ sync mark layer-wise gains after RL.

	L10	L12	L14	L16	L18	L20	L22	L24	L26	L28	L30
1.7B SFT sync	0.232	0.185	0.178	0.194	0.235	0.251	0.231	0.246	**0.284**	–	–
1.7B RL sync	0.263	0.228	0.239	0.299	**0.300**	0.288	0.276	0.280	0.290	–	–
1.7B Δ sync	+0.031	+0.043	+0.061	**+0.105**	+0.065	+0.036	+0.044	+0.034	+0.006	–	–
4B SFT sync	0.210	0.189	0.160	0.168	0.149	0.162	0.161	0.149	**0.215**	0.193	0.175
4B RL sync	0.213	0.194	0.167	0.176	0.157	0.171	0.171	0.162	**0.224**	0.207	0.195
4B Δ sync	+0.003	+0.005	+0.007	+0.007	+0.009	+0.009	+0.011	+0.013	+0.008	+0.014	**+0.020**

C.3 Qualitative Examples and Thread Summaries

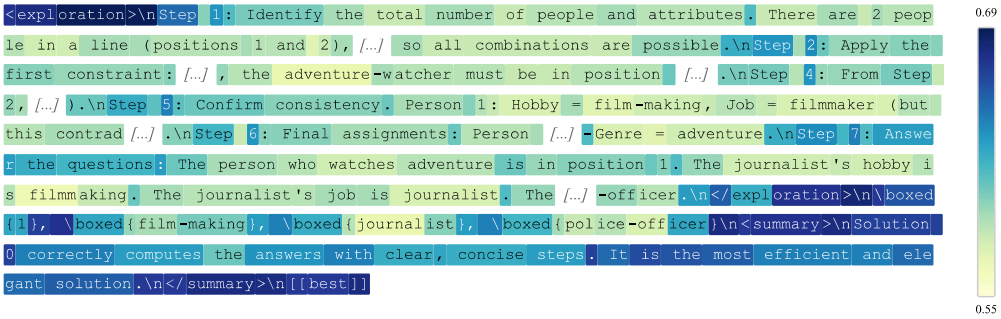


Figure 8: A detailed token-level view of cross-thread intensity on a selected BEST thread from Lattice-1.7B on a LiveBench problem. Darker shading indicates stronger cross-thread interaction, with the strongest signals occurring near exploration and self-assessment tokens.

By inspecting the generated reasoning traces from LACE models, we observe several emergent behaviors that suggest enhanced collaborative reasoning capabilities. Figure 8 provides a detailed token-level view of cross-thread intensity on a selected BEST thread from Lattice-1.7B on a LiveBench problem.

As shown in Figure 8, cross-thread attention peaks around semantically meaningful positions, especially during exploration and self-assessment. This complements the main-text four-thread visualization in Figure 5, showing more clearly how the selected thread absorbs peer information at the token level.

The summaries in Figure 9 contain explicit comparisons between solutions. Thread 2 describes its approach as similar to the reference, while Thread 4 compares its approach with Solution 2 and judges its own steps to be more verbose and less efficient.

Thread 1 Summary

Solution 0 correctly finds the bases 21 and 49 by analyzing the division and checking divisors of 56. It is the most direct and efficient solution.

BEST

Thread 2 Summary

This solution follows a similar approach to the reference but with slightly different phrasing. It is correct and efficient.

SUCCESS

Thread 3 Summary

This solution uses modular arithmetic and the divisibility property to derive the same result as the reference method. It is concise and efficient.

SUCCESS

Thread 4 Summary

Solution 3 follows a similar algebraic approach as Solution 2 but with slightly more verbose steps. It is correct but less efficient than Solution 2.

SUCCESS

Figure 9: Self-assessment summaries from four parallel threads on an AIME 25 problem. Thread 1 correctly identifies itself as the best solution, while other threads acknowledge their relative positions. Displayed threads 1–4 correspond to identifiers 0–3 in the original traces and summaries.

D Analytical Results

We compare candidate coverage under independent sampling and LACE using a conditional-error model. Fix a problem X, and let $\mathcal{C} \subseteq \mathcal{Y}$ be the set of correct reasoning outcomes in a discrete outcome space $\mathcal{Y}$. All probabilities below are conditional on X, which we omit for readability. Let P_{base} be the baseline distribution over outcomes, with error rate $\epsilon_0 = P_{\text{base}}(y \notin \mathcal{C})$. We consider $N \geq 2$ threads and $0 < \epsilon_0 < 1$. Define $S = 1$ when at least one thread returns a correct outcome and $S = 0$ when all threads fail.

Lemma 1 (Independent Failure Probability). For N outcomes $y_1, \ldots, y_N$ sampled independently from P_{base}, the probability that none is correct is

$$P_{\text{IND}}(S = 0) = \prod_{k=1}^{N} P_{\text{base}}(y_k \notin \mathcal{C}) = \epsilon_0^N. \tag{3}$$

This follows directly from independence. Thus, independent sampling has candidate coverage $P_{\text{IND}}(S = 1) = 1 - \epsilon_0^N$.

Theorem 1 (Conditional Coverage Improvement). For LACE, let $E_k = \{y_k \notin \mathcal{C}\}$ denote failure of thread k, and let $E_{<k} = \bigcap_{i=1}^{k-1} E_i$. Thread indices specify an order for the probability factorization; the threads may still interact concurrently during generation. Assume that the first thread has the same marginal failure probability as the baseline, $P_{\text{LACE}}(E_1) = \epsilon_0$. For every $k \in \{2, \ldots, N\}$ with $P_{\text{LACE}}(E_{<k}) > 0$, assume

$$P_{\text{LACE}}(E_k \mid E_{<k}) = \epsilon_0 - \delta_k, \qquad 0 \leq \delta_k \leq \epsilon_0. \tag{4}$$

Then

$$P_{\text{LACE}}(S = 0) \leq \epsilon_0^N, \qquad P_{\text{LACE}}(S = 1) \geq P_{\text{IND}}(S = 1). \tag{5}$$

Both inequalities are strict if $\delta_k > 0$ for at least one index k whose conditioning event has positive probability.

Proof. If $P_{\text{LACE}}(E_{<k}) = 0$ for some $k \in \{2, \ldots, N\}$, then the all-thread failure event, which is a subset of $E_{<k}$, also has probability zero. Since $\epsilon_0 > 0$, this gives strict improvement over independent sampling without conditioning on a zero-probability event.

Otherwise, all conditioning events have positive probability. The chain rule and the assumptions in Equation (4) give

$$\begin{aligned} P_{\text{LACE}}(S = 0) &= P_{\text{LACE}}(E_1) \prod_{k=2}^{N} P_{\text{LACE}}(E_k \mid E_{<k}) \\ &= \epsilon_0 \prod_{k=2}^{N} (\epsilon_0 - \delta_k). \end{aligned} \tag{6}$$

Each factor $\epsilon_0 - \delta_k$ lies in $[0, \epsilon_0]$, establishing Equation (5). If $\delta_k > 0$ for at least one $k \in \{2, \ldots, N\}$, then

$$\epsilon_0 \prod_{k=2}^{N} (\epsilon_0 - \delta_k) < \epsilon_0^N \quad \Longrightarrow \quad P_{\mathrm{LACE}}(S = 1) > P_{\mathrm{IND}}(S = 1). \tag{7}$$

□

Interpretation and scope. An idealized rejection sampler illustrates how avoiding incorrect outcomes can increase coverage. Suppose it excludes a set $\mathcal{M} \subseteq \mathcal{Y} \setminus \mathcal{C}$ of incorrect outcomes with baseline probability mass $\pi = P_{\mathrm{base}}(\mathcal{M})$, where $0 < \pi \leq \epsilon_0 < 1$. Renormalizing the remaining distribution gives $Q(\mathcal{C}) = P_{\mathrm{base}}(\mathcal{C})/(1 - \pi) > P_{\mathrm{base}}(\mathcal{C})$. This is an illustrative mechanism for reducing failure probability. For LACE, Equation (4) remains a modeling assumption whose validity must be assessed empirically. The result concerns candidate coverage (Pass@N); the accuracy of the final self-selected answer additionally depends on the selection mechanism.

E Broader Impacts

LACE may improve the reliability and efficiency of reasoning models by allowing multiple solution paths to coordinate during generation. Potential benefits include stronger mathematical reasoning, planning, and interactive problem solving. At the same time, stronger reasoning and agentic capabilities may also be dual-use, including possible misuse for automated problem solving in harmful settings. Responsible deployment should therefore follow standard model safety practices, including careful evaluation, access controls when appropriate, and monitoring for misuse.

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: The abstract and introduction state the paper's scope as a lattice-attention framework for cross-thread reasoning, and the claims are supported by the method, experiments, and ablations in Sections 3.1 to 3.3 and 4.1.

   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: Section 6 discusses communication overhead, output-format instability beyond the mathematical RL training domain, and the need for domain-specific selection criteria and adaptation to open-domain agent tasks. The overhead measurements are reported in Section B.4 and Tables 11 and 12.

   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: [Yes]

Justification: Section D states the conditional-error assumptions used in the analytical comparison between independent sampling and LACE and provides the corresponding derivation. This result is intended as an explanatory analysis rather than a fully general guarantee.

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: The architecture, data curation procedure, training objectives, baselines, and evaluation metrics are described in Sections 3.1 to 3.3 and 4.1. Section A provides model initialization, training and evaluation settings, and data statistics.

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: [No]

   Justification: The camera-ready paper and supplementary prompt templates document the architecture, training recipe, data curation procedure, and evaluation protocol. They do not include a public code and curated-data release with complete reproduction instructions.

   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: The experimental setup is described in Sections A and 4.1, including model initialization, baselines, datasets, thread count, training stages, optimizer, learning rates, reward variants, rollout engines, checkpoint choices, training seed, and evaluation settings.

   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: Table 1 reports mean $\pm$ standard deviation over at least three evaluation runs per benchmark. These standard deviations summarize evaluation variability for fixed trained checkpoints; the training seed is fixed to 42, as described in Section A.2.

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: Section A describes the training frameworks, rollout engines, checkpoint steps, and relative GPU and generation-length settings. Section B.4 reports the decoding hardware and memory, latency, and FLOPs measurements.

   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: The work uses public benchmarks and model/data resources for machine learning research, and we are not aware of any deviation from the NeurIPS Code of Ethics. It does not involve research with human subjects or collection of private personal data.

   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: [Yes]

    Justification: Section E discusses potential benefits for reliable reasoning and possible dual-use risks from stronger reasoning and agentic capabilities. It also notes that responsible deployment should follow safety evaluation, access-control, and misuse-monitoring practices where appropriate.

    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: The camera-ready materials do not release a high-risk pretrained model, image generator, scraped dataset, or other asset requiring special release safeguards. Potential misuse and responsible deployment considerations are discussed in Section E.

    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: [No]

Justification: The paper credits the existing models, datasets, and benchmarks, including Qwen3, LLaMA-3.1, DAPO, DART-Math, AIME, LiveBench, HotpotQA, TALES/TextWorld, and WebShop. It does not provide a complete inventory of license names and terms of use for all assets.

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: [N/A]

Justification: The camera-ready materials do not include a standalone release of new datasets, checkpoints, or software artifacts. The paper and supplementary prompt templates document the proposed method and data curation procedure.

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: The work does not involve crowdsourcing experiments or 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: The work does not involve crowdsourcing experiments or research with human subjects, so IRB approval or equivalent review is not applicable.

    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: [Yes]

    Justification: LLMs are central to the research: LACE modifies Qwen3 and LLaMA-3.1 backbones, and the data curation pipeline uses Qwen3-235B-A22B to generate diverse multi-thread reasoning traces. The method and model initialization are described in Sections A, 3.1 and 3.2, and the supplementary prompt templates document data generation. Use of LLMs for writing or editing is separate from the scientific methodology.

    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.