Recursive Multi-Agent Systems

Jiaru Zou[1] **Rui Pan**[2] **Ruizhong Qiu**[2] **Pan Lu**[1] **Shizhe Diao**[3] **Jindong Jiang**[3]
Hanghang Tong[2] **Tong Zhang**[2] **Markus J. Buehler**[4] **Jingrui He**[2†] **James Zou**[1†]

[1]Stanford University [2]UIUC [3]NVIDIA [4]MIT

Project Page: https://recursivemas.github.io

Abstract

Recursive or looped language models have recently emerged as a new scaling axis by iteratively refining the same model computation over latent states to deepen reasoning. We extend such scaling principles from a single model to multi-agent systems, and ask: *Can agent collaboration itself be scaled through recursion?* To this end, we introduce **RecursiveMAS**, a recursive multi-agent framework that casts the entire system as a unified latent-space recursive computation. RecursiveMAS connects heterogeneous agents as a collaboration loop through the lightweight RecursiveLink module, enabling in-distribution latent thoughts generation and cross-agent latent state transfer. To optimize our framework, we develop an inner-outer loop learning algorithm for iterative whole-system co-optimization through shared gradient-based credit assignment across recursion rounds. Theoretical analyses of runtime complexity and learning dynamics establish that RecursiveMAS is more efficient than standard text-based MAS and maintains stable gradients during recursive training. Empirically, we instantiate RecursiveMAS under 4 representative agent collaboration patterns and evaluate across 9 benchmarks spanning mathematics, science, medicine, search, and code generation. In comparison with advanced single/multi agents and recursive computation baselines, RecursiveMAS consistently delivers an average accuracy improvement of 8.3%, 1.2×-2.4× end-to-end inference speedup, and 34.6%-75.6% token usage reduction.

1 Introduction

To tackle complex tasks, a single language model often falls short due to limited capacity, myopic generation, or inefficient exploration of the solution space [1, 2, 3]. Once intelligence reaches a threshold, a natural direction is to treat individual models as specialized agents and organize them as a collaborative system [4, 5]. A multi-agent system (MAS) [6, 7] can scale performance by enabling individuals to work together and contribute complementary strengths. Consider a set of heterogeneous agents, each assigned a distinct role and expertise. The system can either arrange agents into a sequential pipeline [8, 9] to progressively decompose and solve a problem, or engage and integrate multiple domain-specialized agents for the task [10, 11, 12].

While MAS establishes a structural foundation, the next question is how to enable the system to evolve over time and adapt to different scenarios. Prior work has explored prompt-based adaptation [13, 14, 15], where model interactions are improved through the iterative refinement of shared context. Although these updated prompts can help agents generate more aligned responses to the question, each agent itself cannot improve. A more principled line of work is to optimize agents through learning [16, 17, 18]. However, training entire agents inside the system is hard, as updating all model

†Corresponding Authors. Contact: {jiaru,jamesz}@stanford.edu.

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

parameters is non-trivial [19], and the sequential dependency in text-based interactions introduces substantial latency when agents must wait for others to complete generation.

Instead of improving each agent's capabilities as a standalone component, we adopt a higher-level learning perspective and aim to co-evolve and scale the entire system as an integrated whole. We recast agent collaboration through the lens of recursive language models (RLMs) [20, 21, 22], where a shared set of layers is iteratively applied and optimized within a continuous latent space. In this view, the entire system can be treated as a recursive computation, where each agent acts like an RLM layer, iteratively passing latent representations to the next and forming a looped interaction process.

We call this new system-level agentic recursion framework **RecursiveMAS**. Without updating all model parameters, agents are connected and iteratively optimized solely via the lightweight **RecursiveLink**, a two-layer residual projection module for latent states transmission and refinement. An *inner RecursiveLink* within each agent first consolidates the model's ongoing latent thoughts between input and output spaces during auto-regressive generation. An *outer RecursiveLink* then bridges hidden representations across heterogeneous agents (different model types and sizes), enabling seamless cross-agent interaction. Together, all agents are chained in a unified loop to perform iterative latent collaboration, with only the last agent producing the textual output in the final recursion round.

Correspondingly, we pair RecursiveMAS with an **Inner-Outer Loop** training paradigm for progressive co-optimization. The *inner loop* provides a preliminary model-level warm start for each agent, by training its inner RecursiveLink to better align with latent thoughts generation. The *outer loop* then trains the outer RecursiveLink across agents at the system-level, with gradients recursively back-propagated through the full computation traces over recursion rounds. By exposing each agent to the feedback of itself and others from previous rounds, RecursiveMAS learns the iterative refinement of multi-agent collaboration, thus enabling the entire system to optimize in a unified manner.

To justify why recursion should occur in latent space rather than text-mediated interaction, we provide two theoretical analyses on runtime complexity and learning dynamics. From an architectural standpoint, RecursiveLink enables direct transformation of latent-space information, avoiding repeated decoding of intermediate agents with more efficient runtime complexity. From the learning perspective, latent-space connections in RecursiveMAS maintain stable gradient propagation flow across recursion rounds during training, avoiding the gradient vanishing induced by text-based interactions.

Empirically, we evaluate RecursiveMAS on 9 benchmarks spanning mathematics, science, medicine, search, and code generation. We instantiate RecursiveMAS with diverse model families, including Qwen3/3.5 [23], LLama-3 [24], Gemma3 [25], and Mistral [26, 27], and adapt our framework to 4 representative MAS collaboration scenarios: step-by-step sequential reasoning, mixture-of-experts collaboration, expert-to-learner knowledge distillation, and tool-integrated deliberation. In comparison with advanced recursive language models and MAS baselines, RecursiveMAS achieves an average accuracy improvement of 8.3%, while delivering 1.2×–2.4× inference speedup and reducing token usage by 34.6%–75.6%. In addition, RecursiveMAS is structure-agnostic and can generalize to various agent collaboration patterns with effective performance. Detailed analyses of scaling laws with deeper recursion, RecursiveLink architectures, semantic distributions across recursions, and training cost further validate the efficiency and performance scalability of RecursiveMAS.

2 Preliminary

Auto-regressive Generation in Latent Space. Let $f_\theta(\cdot)$ denote a standard Transformer model [28] parameterized by θ. Given a question x with input embeddings $E = [e_1, \ldots, e_t] \in \mathbb{R}^{t \times d_h}$, where d_h is the hidden dimension, the model performs the forward pass through its residual stream to compute the last-layer hidden state h_t at step t. In standard text generation, h_t is decoded to the vocabulary space to predict the next token. In contrast, latent generation keeps the auto-regression entirely in continuous high-dimensional vector space by directly feeding the previous latent embeddings h_t back into the next forward pass. Formally, the next latent generation at step $t+1$ is:

$$h_{t+1} = f_\theta([E_{\leq t}; h_t]). \tag{1}$$

We refer to the newly generated latent states (e.g., h_{t+1}) as the model's ongoing *latent thoughts*.

Recursive Computation. A recursive language model (RLM) increases reasoning depth by reusing the same transformation across recurrent steps. Consider a Transformer f_θ with L layer blocks,

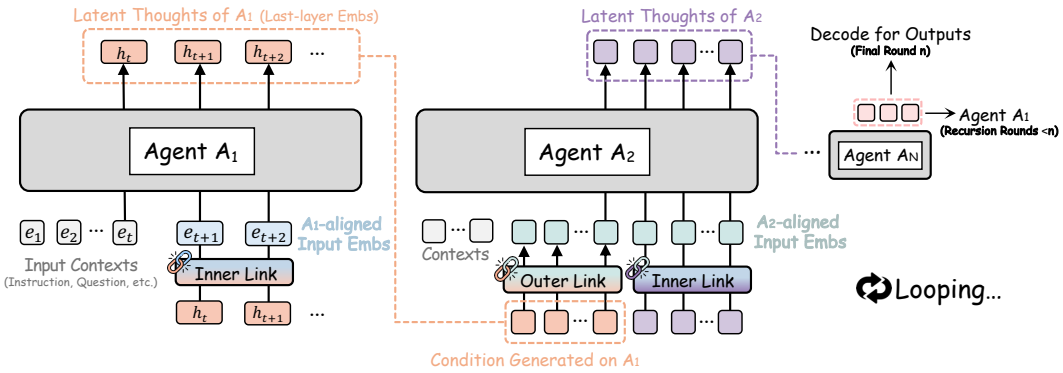

Figure 1: **Overall Architecture of RecursiveMAS.** Each agent first leverages the inner RecursiveLink to perform latent thoughts generation, and then transfers the generated information to the next agent through the outer RecursiveLink. After the last agent finishes generation, its latent thoughts are fed back to the first agent, thereby forming a recursive loop within the multi-agent system.

denoted as $f_\theta = \mathcal{M}_L \circ \cdots \circ \mathcal{M}_1$. Instead of passing the input through the L-layer stack only once to obtain output embeddings, an RLM reuses the same stack for n times of forward iterations, i.e.,

$$H^{(0)} = E, \quad H^{(r)} = f_\theta\big(H^{(r-1)}\big), \quad r = 1, \ldots, n. \tag{2}$$

The last round of latent representation $H^{(n)}$ is obtained through recursive refinement over the same shared Transformer layers, and is subsequently used for the final prediction.

LLM-based Multi-Agent Evolution. We define a multi-agent system $\mathcal{S}$ [4, 29] composed of N agents denoted as $\mathcal{A} = \{A_1, \ldots, A_N\}$, where each LLM agent A_i corresponds to f_{θ_i} with its own last-layer representations H_i. We then denote the collective latent state of the system by $\mathcal{H} = \{H_1, \ldots, H_N\}$. Given any input problem x with the ground-truth y, the system $\mathcal{S}$ orchestrates interactions among agents to collaboratively produce a final prediction. With this setup in place, we now formalize the evolution of agents under recursive computation.

Definition 2.1: Recursive Multi-Agent Evolution

*A **recursive evolution** is the progressive refinement of $\mathcal{H}$, where each agent adjusts its latent representation through iterative interaction with others and its own reasoning state, so that the updated system is better aligned for the given problem, i.e.* $\mathcal{S}^{(0)} \xrightarrow[\text{Evolve}]{H^{(1)}} \mathcal{S}^{(1)} \xrightarrow[\text{Evolve}]{H^{(2)}} \cdots \xrightarrow[\text{Evolve}]{H^{(n)}} \mathcal{S}^{(n)}$.

Collaboration Pattern. As MAS architectures are generally not fixed and can vary across tasks, we do not restrict the collaboration pattern to a single style. In this paper, we consider 4 commonly adopted collaboration patterns in multi-agent systems: (i) *Sequential Style*, where we follow the chain-of-agents setting [30] to assign three agents with complementary roles of `Planner`, `Critic`, and `Solver` and progressively decompose, judge, and solve the problem; (ii) *Mixture Style*, where a mixture of domain-specialized agents (`Math`, `Code`, `Science`) reasons over the input problem in parallel, and their outputs are aggregated by a `Summarizer` agent to form the final answer; (iii) *Distillation Style*, where a larger, more capable `Expert` agent is paired with a smaller, faster `Learner` agent to distill expert knowledge while retaining higher generation efficiency; and (iv) *Deliberation Style*, where an inner-thinking `Reflector` is paired with a `Tool-Caller` that can invoke external tools (e.g., Python or search APIs). The agents iteratively exchange, critique, and refine candidate solutions until reaching a shared consensus, after which the `Tool-Caller` produces the final answer.

3 Building a Recursive Multi-Agent System

We introduce RecursiveMAS, an end-to-end recursive framework that links heterogeneous LLM agents together to scale the entire system through efficient and seamless latent collaboration. In the following, we will first elaborate the detailed architectural design of RecursiveMAS, and then present the corresponding recursive learning algorithm. We also interleave theoretical analyses throughout the method pipeline to support underlying design principles.

3.1 A Lightweight RecursiveLink

A language model's last-layer hidden states provide a natural representation of its generated semantics. The RecursiveLink $\mathcal{R}$ is designed to preserve and transmit this information from one embedding space to another. In RecursiveMAS, the transition arises in two cases: (i) *Dense-to-Shallow Transition*, where the previous step's last-layer embeddings are fed back as the next-step input embeddings during latent thoughts generation; and (ii) *Cross-Model Transition*, where one model's newly generated latent representations are passed as conditioning inputs to another. As illustrated in Figure 2, we bridge these two transitions through the inner and outer links.

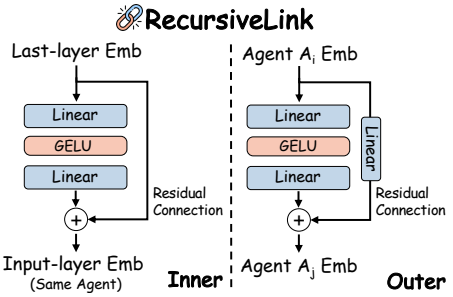

Figure 2: Illustration on the Inner (*left*) and Outer (*right*) RecursiveLink Designs.

Inner Link. Each LLM agent $A_i \in \mathcal{A}$ is paired with its own inner RecursiveLink $\mathcal{R}_{\text{in}}$ during auto-regressive latent generation. Given any new last-layer embedding vector h, $\mathcal{R}_{\text{in}}$ transforms it as:

$$\mathcal{R}_{\text{in}}(h) = h + W_2\,\sigma(W_1 h), \tag{3}$$

where W_1 and W_2 are two standard linear layers, $\sigma(\cdot)$ is the GELU activation, and the residual connection preserves the original latent semantics. The transformed embedding from the inner link $\mathcal{R}_{\text{in}}$ is then used as the input to the next step forward pass of agent A_i.

Outer Link. An outer RecursiveLink $\mathcal{R}_{\text{out}}$ connects each of two heterogeneous agents with different hidden dimensions. To support this, an additional linear layer W_3 is introduced in the residual branch to map the source embedding from agent A_i into the target embedding space of agent A_j, i.e.,

$$\mathcal{R}_{\text{out}}(h) = W_3 h + W_2\,\sigma(W_1 h). \tag{4}$$

Why Residual Connection? The residual branch largely preserves the original semantics of the input, allowing the RecursiveLink network to focus on *aligning distributional differences* rather than learning the full projection from scratch. This leads to more stable and efficient training. We also explore other alternatives and empirically validate our proposed design in Section 5.

3.2 Chain All Agents Together as a Loop

In recursive language models (RLMs), Transformer layers are connected through hidden states, and the residual stream loops across these layers to increase reasoning depth. Under this view, we cast each agent in RecursiveMAS as an RLM layer, with information flowing and recurring within and across agents as the hidden stream of the system. As shown in Figure 1, each agent contributes by reasoning and interacting with others in the latent space, together forming a recursive loop.

Latent Thoughts Generation inside Agents. We start by describing how each agent unfolds reasoning through the auto-regressive generation of latent thoughts. Specifically, given input contexts' embeddings $E_{A_1} = [e_1, e_2, \ldots, e_t]$ for the question and the agent-specific instructions, the first agent A_1 passes E_{A_1} through the Transformer and computes the last-layer hidden representation h_t at step t. Then, we insert h_t into the inner link $\mathcal{R}_{\text{in}}$ to map the distribution back into the input embedding space for the next step, yielding $e_{t+1} = \mathcal{R}_{\text{in}}(h_t)$. Agent A_1 repeats this process auto-regressively for m forward steps, generating a new continuous sequence of latent thoughts $H_{A_1} = [h_t, h_{t+1}, \ldots, h_{t+m}]$.

Interaction across Heterogeneous Agents. After the completion of agent A_1, its latent thoughts H_{A_1} are sent to the second agent A_2 for cross-agent interaction. To achieve seamless information transmission across different agent types, we pass H_{A_1} through the outer link $\mathcal{R}_{\text{out}}$ to transform it into input embeddings aligned with agent A_2. Next, agent A_2 starts latent thoughts generation conditioned on (i) its own input contexts and (ii) the transferred latent thoughts from A_1 (i.e., $E_{A_2} \oplus \mathcal{R}_{\text{out}}(H_{A_1})$).

We continue this interaction process across all N consecutive agents in RecursiveMAS. At the end of the agent chain, once the last agent A_N completes latent thoughts generation, its latent outputs (carrying the system's overall latent answer representations) are passed back to the first agent A_1 through the RecursiveLink, thereby closing the recursive loop. This recurrent connection allows each new recursion round to condition on information produced in previous rounds, so that each agent can iteratively reflect on earlier system outputs and refine their current generation. Throughout

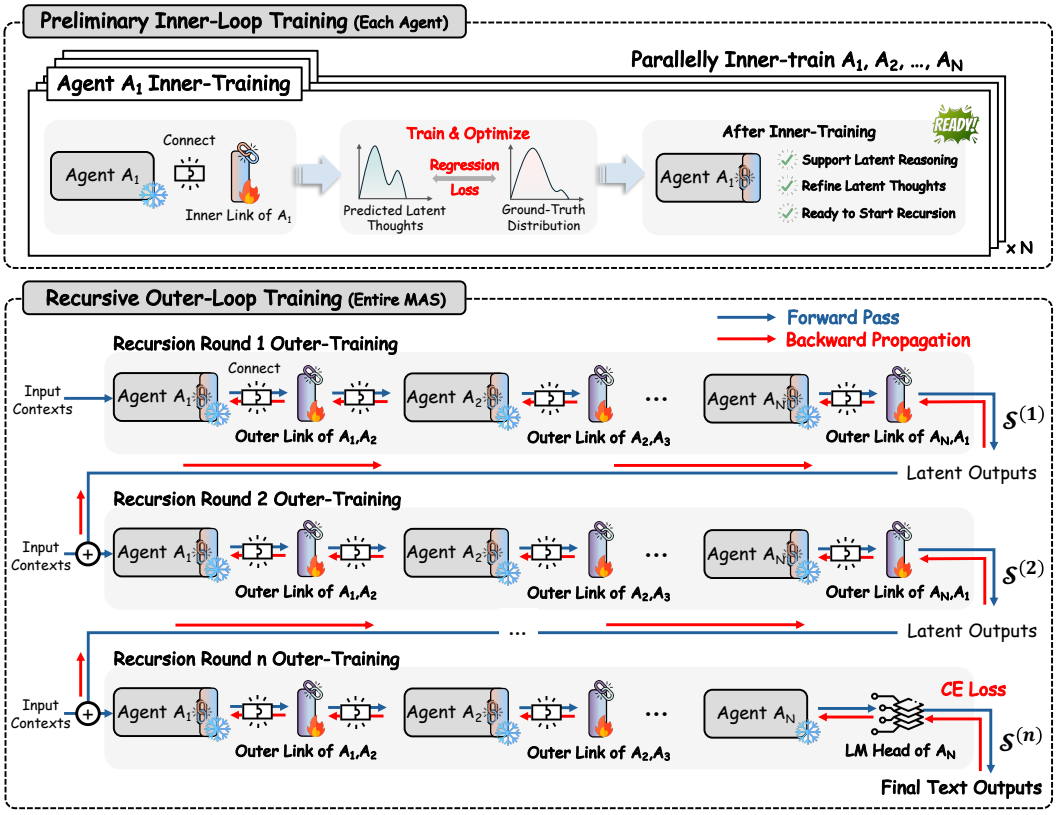

Figure 3: **Inner-Outer Training Pipeline of RecursiveMAS.** We first perform *inner-loop training* for each agent in parallel to warm up the inner RecursiveLink for latent thoughts generation, and then conduct *outer-loop training* to recursively optimize the outer RecursiveLink over the entire system.

intermediate recursion rounds, all agents collaborate entirely in the latent space. Only after the final recursion round, the agent A_N decodes the textual output as the system's final answer to the question.

End-to-End Complexity Analyses. To characterize the architectural efficiency of the RecursiveMAS pipeline, we next provide a theoretical analysis on its end-to-end running time complexity. The following proposition compares RecursiveMAS with the text-based recursive multi-agent system, in which agents follow the same multi-round recursive collaboration structure but communicate through an explicit text medium rather than the latent-space interaction enabled by the RecursiveLink.

Proposition 3.1 (RecursiveMAS Runtime Complexity). *Without RecursiveLink, a text-based Recursive MAS with the same collaboration structure requires runtime complexity of* $\Theta(N(m|V|d_h + (t+m)d_h^2 + (t+m)^2 d_h))$*; In contrast, with RecursiveLink-enabled collaboration, RecursiveMAS achieves an end-to-end runtime complexity of* $\Theta(N(md_h^2 + (t+m)d_h^2 + (t+m)^2 d_h))$.

Remark 3.2. As $d_h \ll |V|$ in practice, RecursiveMAS replaces the expensive per-step vocabulary-space decoding cost $m|V|d_h$ with a much more efficient latent-space transformation md_h^2.

Proposition 3.1 shows the end-to-end runtime advantage of RecursiveMAS. The full proof is provided in Appendix A.1. We also empirically analyze the efficiency advantage of our method in Section 5.

4 Learning to Recur as a Whole

With the framework in place, we next present the recursive learning algorithm, which only needs to train on the RecursiveLink to enable co-optimization of the entire system loop. As illustrated in Figure 3, the learning procedure consists of two stages: (i) a preliminary *inner-loop* to equip

each agent with stronger latent thoughts generation capabilities; and (ii) an iterative *outer-loop* to progressively optimize the system as one unified entity over recursion rounds.

Model-Level Inner-Loop Training. For practical deployment of RecursiveMAS, we directly adopt off-the-shelf text-generation models as agents. To adapt these agents to the latent thoughts generation pattern, we first warm-start them through the inner RecursiveLink $\mathcal{R}_{\text{in}}$. Specifically, given each agent $A_i \in \mathcal{A}$ with parameters θ_i and the training example $(x, y) \in \mathcal{D}_{\text{train}}$, we construct the target latent thoughts distribution by passing the ground-truth text y through the standard input embedding layer Emb_{θ_i} of agent A_i. The objective of training the inner link $\mathcal{R}_{\text{in}}$ corresponding to A_i formulates as:

$$\mathcal{L}_{\text{in}} = 1 - \cos\big(\mathcal{R}_{\text{in}}(H), \text{Emb}_{\theta_i}(y)\big), \tag{5}$$

where H denotes the last-layer latent thoughts generated by agent A_i, and $\cos(\cdot, \cdot)$ denotes the standard cosine similarity. The regression objective here encourages each agent to leverage its inner link $\mathcal{R}_{\text{in}}$ to align latent thoughts with the semantic distribution from the input embedding layer, while eliminating the process of explicit decoding and re-encoding.

System-Level Outer-Loop Training. Next, we iteratively co-optimize the entire system through the outer RecursiveLink $\mathcal{R}_{\text{out}}$. Let $\mathcal{S}^{(r)}$ denote the system state at recursion round $r = 1, \ldots, n$. During outer-loop training, the system is first unrolled along its looped structure for n forward rounds. After the final textual prediction is produced in the last recursion round, we jointly optimize all outer links that connect the system with the following cross-entropy (CE) objective:

$$\mathcal{L}_{\text{out}} = \text{CE}\Big(\mathcal{S}^{(n)}\big(\mathcal{S}^{(n-1)}(\cdots \mathcal{S}^{(1)}(x))\big),\, y\Big). \tag{6}$$

Throughout training, the computation graph is preserved along the full recursive paths. Gradient backpropagation assigns each outer link a shared credit signal according to its global contribution to the final prediction, thereby enabling information flow to be iteratively optimized as a whole.

Learning Advantage of RecursiveMAS. To better understand why latent collaboration of agents in the inner-outer loop training confers a stronger learning advantage, we provide a detailed theoretical analysis below of the gradient propagation process throughout recursive training of RecursiveMAS.

Theorem 4.1 (Gradient Stability). *Under the Realistic Assumptions (stated in Appendix A.2), if tokens are confident with entropy $\leq \epsilon$, where typically $\epsilon \ll 1$: directly applying text-based SFT (denoted by $\mathcal{R}_{text}(h)$) during recursion suffers from gradient vanishing (i.e., gradient norm close to 0); while RecursiveMAS with the RecursiveLink $\mathcal{R}$ maintains stable and near constant gradients (i.e., gradient norm close to 1) during looped backpropagation process. Formally, with probability $\geq 1 - \delta$,*

$$\left\| \frac{\partial \mathcal{R}_{text}(h)}{\partial h} \right\|_2 \leq O(\epsilon) \ll 1, \qquad \left\| \frac{\partial \mathcal{R}(h)}{\partial h} \right\|_2 \geq \Omega\left(1 - \sqrt{\frac{1}{d_h} \log \frac{1}{\delta}}\right). \tag{7}$$

The full proof is provided in Appendix A.2. Theorem 4.1 demonstrates the learning advantage of RecursiveMAS, by allowing gradients to remain informative across recursion rounds. Together, theoretical justifications in Proposition 3.1 and Theorem 4.1 motivate our design of latent-based interaction among agents rather than text mediation, as it makes the whole-system co-optimization of RecursiveMAS easier and more effective. During inference, RecursiveMAS performs recursive generation by following the same n recursion rounds as in the outer-loop training.

5 Empirical Evaluations

Tasks and Datasets. We conduct comprehensive evaluations of RecursiveMAS on nine benchmarks across various domains: (i) *Mathematical Reasoning*, including MATH500 [31], AIME2025 [32], and AIME2026 [33]; (ii) *Scientific and Medical Tasks*, including GPQA-Diamond [34] and MedQA [35]; (iii) *Code Generation*, including LiveCodeBench-v6 [36] and MBPP Plus [37]; and (iv) *Search QA*, including HotpotQA [38] and Bamboogle [39]. We adopt the standard evaluation metric for each dataset. For AIME2025/2026, we report Pass@10 accuracy for testing robustness. Additional benchmark and metrics details are in Appendix B.1.

Models and Baselines. We instantiate RecursiveMAS with diverse agent collaboration patterns, including (i) *Sequential Style*, (ii) *Mixture Style*, (iii) *Distillation Style*, and (iv) *Deliberation Style*,

Table 1: **Performance Scales of RecursiveMAS across Different Recursion Rounds.** We report the accuracy (%, **"Acc."**), end-to-end runtime (s / run, **"Time"**), and overall token usage (**"Token"**). The average standard deviation of RecursiveMAS across 5 independent runs is ± 0.0041 for accuracy, ± 26 for runtime, and ± 33 for tokens. Across all metrics, RecursiveMAS's performance advantages become increasingly significant as the recursion round r increases, with improvements **highlighted**.

Method	Metric	Math500		AIME2025		AIME2026		GPQA-D		MedQA		Code Gen.		Improve
		Light	Scaled	Light	Scaled	Light	Scaled	Light	Scaled	Light	Scaled	Light	Scaled	
Recursive Round r=1														
Recursive-TextMAS	Acc.	71.9	84.2	24.0	71.3	16.7	76.7	28.1	61.5	29.0	76.1	30.7	38.5	Base
	Time	1368	2401	2380	8462	2216	9376	1056	2190	1555	1522	976	8867	Base
	Token	1185	1471	2993	9397	2754	8854	2084	3693	2382	1427	1146	3154	Base
RecursiveMAS	Acc.	75.8	86.3	30.7	80.0	17.3	82.7	30.3	63.1	30.3	78.2	35.1	40.1	**↑ 3.4**
	Time	825	1701	1829	7784	1788	8134	586	1965	1194	1348	449	7908	**×1.2**
	Token	523	816	1622	6338	1576	7021	829	2675	1369	964	577	2198	**↓ 34.6%**
Recursive Round r=2														
Recursive-TextMAS	Acc.	72.5	84.4	23.3	70.7	10.0	77.3	28.7	59.1	28.3	76.1	30.0	38.0	Base
	Time	2204	3958	4247	14380	3960	14110	1825	4207	3097	2745	1847	14792	Base
	Token	2117	2794	5318	16372	4982	16213	3708	6128	4436	2609	1998	5369	Base
RecursiveMAS	Acc.	76.6	87.1	33.3	86.0	18.7	84.0	32.3	64.6	31.2	78.3	36.9	41.3	**↑ 6.0**
	Time	1096	1974	2367	8178	2263	8965	752	2342	1427	1664	627	8329	**×1.9**
	Token	495	953	1614	5314	1552	6657	813	2521	1383	1008	531	2020	**↓ 65.5%**
Recursive Round r=3														
Recursive-TextMAS	Acc.	69.1	85.8	18.0	73.3	16.7	74.7	28.7	58.6	28.5	77.1	29.3	36.5	Base
	Time	2952	6010	6183	19304	5907	19678	3322	7537	4684	3922	2310	22036	Base
	Token	3059	4100	8645	23651	7813	22915	5820	8091	6307	3731	2676	7078	Base
RecursiveMAS	Acc.	77.8	88.2	34.0	86.7	20.0	86.0	32.6	66.2	31.7	79.3	37.4	42.8	**↑ 7.2**
	Time	1360	2320	2727	8981	2629	9623	861	2638	1704	1912	805	10186	**×2.4**
	Token	519	893	1586	5342	1537	6860	786	2524	1378	1056	595	2247	**↓ 75.6%**

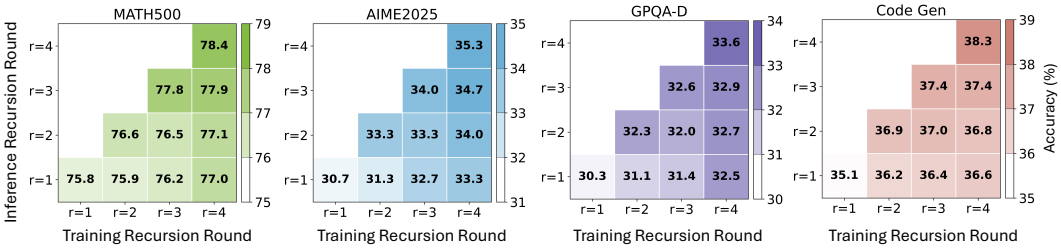

Figure 4: **Performance Landscape of Train-time × Test-time Recursion.** The lightweight RecursiveMAS with sub-1.5B agents shows a clean scaling trend as recursion deepens, with the strongest results consistently appearing in the upper-right as both inference and training recursion deepens.

following the setups described in Section 2. For each collaboration style, we use off-the-shelf LLMs from diverse model families, covering Qwen-3/3.5 [40, 41], Llama-3.1/3.2 [24], Gemma-3 [25], and Mistral [26], to construct heterogeneous agent compositions. Detailed model configurations, scales, and role assignments in RecursiveMAS are provided in Appendix B.2 (Table 3).

For baseline comparisons, we evaluate RecursiveMAS against (i) *Single Advanced Agents*, where individual LLM agents from each collaboration pattern are isolated as standalone models to solve problems, such as the final agent in Sequential Style and each domain specialist in Mixture Style. For fair comparison, we provide full supervised and LoRA fine-tuning [42] for single models on the same training set. (ii) *Recursion-based Methods*, including single recursive language models, LoopLM [22], and Recursive-TextMAS, where agents collaborate in the same way as RecursiveMAS but interact through text instead of latent thoughts; and (iii) additional *Representative Multi-Agent Frameworks*, including TextGrad [43] and Mixture-of-Agents (MoA) [6] for more holistic structure-wide evaluations. Detailed baseline implementations are provided in Appendix B.3.

Training and Implementation Details. For inner-outer loop training, we curate a diverse training set spanning multiple domains, sourced from s1K [44] for mathematical problem solving, m1k [45] for medical and scientific tasks, OpenCodeReasoning [46] for code generation, and ARPO-SFT [47] for agentic tool-augmentation (Python Code/Search-API) settings. We use AdamW with a learning

Table 2: **Whole-System Performance of RecursiveMAS.** Under identical training budgets and matched model setups, RecursiveMAS (with recursion round $r = 3$) consistently outperforms advanced fine-tuned single agents, alternative MAS frameworks, and recursive computation baselines.

Method	MATH500	AIME2025	AIME2026	GPQA-D	LiveCodeBench	MedQA
Single Agent (w/ LoRA)	83.1	70.0	73.3	62.0	37.4	76.1
Single Agent (w/ Full-SFT)	83.2	73.3	76.7	62.8	38.6	77.0
Mixture-of-Agents (MoA)	79.8	60.0	63.3	47.6	27.0	57.5
TextGrad	84.9	73.3	76.7	62.5	39.8	77.2
LoopLM	84.6	66.7	63.3	48.1	24.9	56.4
Recursive-TextMAS	85.8	73.3	73.3	61.6	38.7	77.0
RecursiveMAS	**88.0**	**86.7**	**86.7**	**66.2**	**42.9**	**79.3**

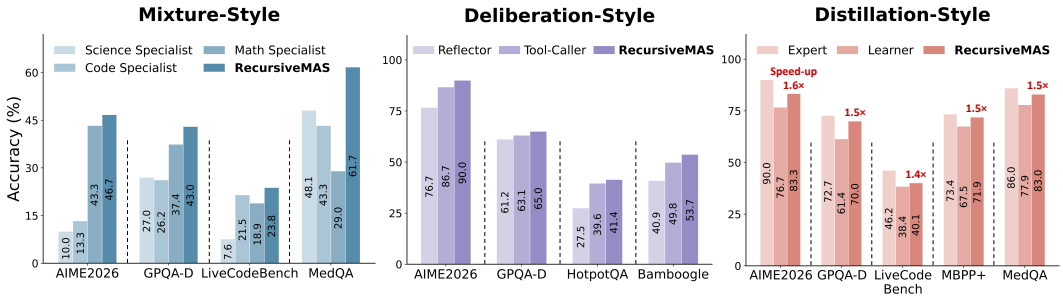

Figure 5: **Generalization of RecursiveMAS across Common MAS Collaboration Patterns.** RecursiveMAS adapts to diverse MAS structures, delivering consistent gains in accuracy and efficiency.

rate of 5e-4, a cosine learning rate scheduler, and a batch size of 4. During inference, we set top-p to 0.95 and use a temperature of 0.6 for most reasoning tasks and 0.2 for code generation, as suggested in each model's official report. We report the mean performance over five independent runs. More training, inference details, and hyperparameter setups are provided in Appendix B.4.

5.1 Scaling Performance via Recursion

We begin by evaluating how RecursiveMAS performs across different recursion depths $r = 1, 2, 3$. As shown in Table 1, we analyze agent collaboration behavior from three complementary metrics. Across all cross-domain tasks, both light and scaled RecursiveMAS in sequential-style exhibit a consistent upward trend as recursion depth increases. When compared with the text-based recursion, RecursiveMAS consistently improves over the baseline by an average of 8.1% at $r = 1$, 19.6% at $r = 2$, and 20.2% at $r = 3$, with performance advantage more pronounced as the recursion deepens. Additionally, under identical MAS architectures, RecursiveMAS delivers steadily increasing efficiency gains across recursion rounds, accelerating end-to-end inference time from $1.2\times$ to $2.4\times$ while reducing output tokens from 34.6% to 75.6%. Figure 4 further examines the scaling behavior of recursion in RecursiveMAS by jointly varying the training-time and inference-time recursion rounds. Increasing inference depth continues to improve systems trained with fewer rounds, while deeper training shifts the entire performance frontier upward. We also conduct detailed case studies of RecursiveMAS across domains to validate the observations above, as provided in Appendix G.

5.2 Whole-System Performance of RecursiveMAS

Table 2 further compares RecursiveMAS at the whole-system level against a broader set of baselines, including single fine-tuned agents, representative multi-agent frameworks, and alternative recursive methods. RecursiveMAS achieves an average improvement of 8.3% over the strongest baseline on each benchmark. Under same training budgets, fine-tuning individual solver agents improves their off-the-shelf counterparts, while RecursiveMAS delivers further 11.7% gains by optimizing cross-agent collaboration at the system level. In addition, RecursiveMAS maintains a clear advantage over other strongest baselines such as TextGrad and LoopLM, especially on reasoning-intensive tasks, with accuracy gains of 18.1% on AIME2025, 13.0% on AIME2026, and 5.4% on GPQA-Diamond.

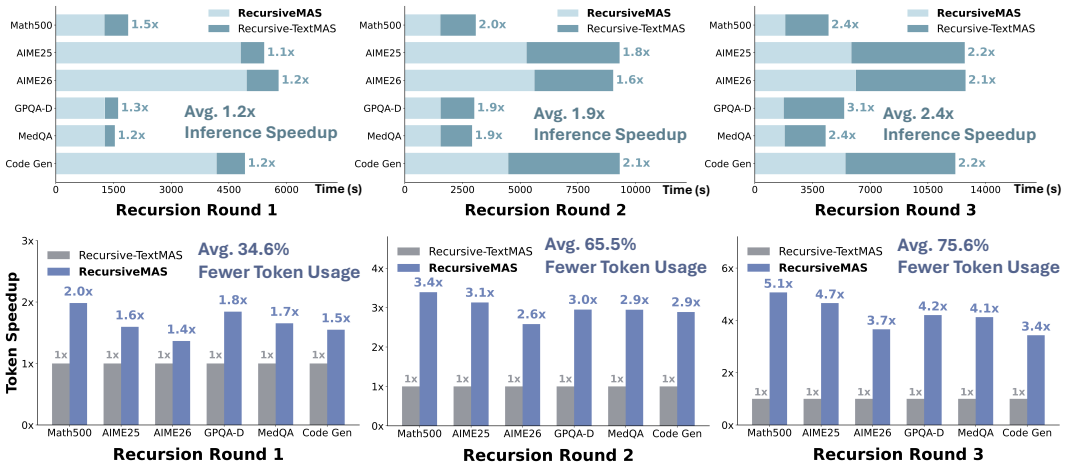

Figure 6: **Efficiency Analyses of RecursiveMAS throughout Recursion.** We analyze RecursiveMAS in terms of (i) end-to-end inference time and (ii) overall token usage. As recursion deepens from round 1 to 3, RecursiveMAS achieves increasing inference speedup and token reductions.

5.3 Can RecursiveMAS Generalize across Different Collaboration Patterns?

Beyond the sequential setting, we also instantiate RecursiveMAS under three additional MAS collaboration patterns in Table 3 to assess whether our method is agnostic to specific system structures and generalizes across diverse usage scenarios. As shown in Figure 5, we compare our recursion framework against strong standalone agents within each collaboration pattern. Across all styles, RecursiveMAS consistently outperforms the strongest standalone agent while preserving inference speed advantage. We leave detailed results on the generalizability of RecursiveMAS in Appendix D.1.

5.4 Efficiency Analyses on Recursive Collaboration

Inference Time Speedup. We next analyze the efficiency of RecursiveMAS to empirically support the complexity advantage stated in Proposition 3.1. We compare against Recursive-TextMAS to study how end-to-end inference scales with recursion depth, as shown in Figure 6 (top). Specifically, RecursiveMAS achieves an average speedup of $1.2\times$ at $r = 1$, and increases to $1.9\times$ and $2.4\times$ at $r = 2$ and $r = 3$, respectively. This trend is consistent with our design: by carrying out recursive collaboration directly in latent space, RecursiveMAS avoids repeated intermediate text generation and exhibits increasingly favorable scaling as recursion deepens.

Overall Token Usage Reduction. Figure 6 (bottom) further shows that RecursiveMAS greatly reduces overall system token usage. While the text baseline incurs rapidly growing token overhead with increasing recursion depth due to repeated intermediate text decoding at every round, RecursiveMAS reduces token usage by 34.6% at $r = 1$, with the reduction further growing to 75.6% at $r = 3$. The efficiency advantage of RecursiveMAS also carries over to training. Additional **training cost analyses** in Appendix D.2 show that RecursiveMAS achieves a better cost-performance trade-off than other training baselines while using fewer trainable parameters and lower GPU memory.

6 In-Depth Analysis on RecursiveMAS

Semantic Representations across Recursion. We analyze how the semantic distribution of RecursiveMAS changes across different recursion rounds. Under the scaled sequential setting of RecursiveMAS, we randomly sample 500 question-answer pairs spanning all downstream domains. We then use the solver agent's input embedding layer to map each ground-truth answer string into embedding representations, which serves as the reference semantic distribution. We run RecursiveMAS at recursion rounds $r = 1, 2, 3$ to generate final answers for all these 500 questions, map the generated answers into embeddings using the same input embedding layer, and visualize both the ground-truth reference ("purple") and newly generated distributions ("orange") via PCA projection. In Figure 7, the generated answers at $r = 1$ remain visibly shifted from the ground-truth distribution, but this discrepancy progressively narrows as depth increases, with the two distributions becoming

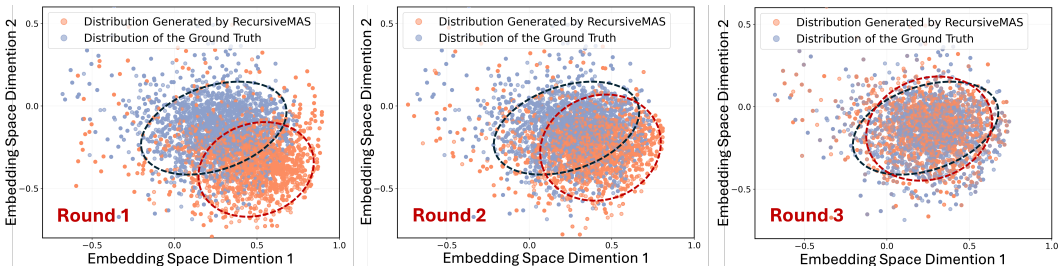

Figure 7: **Semantic Representations of RecursiveMAS across Different Recursion Rounds.** We visualize the semantic distribution of the final answers generated by RecursiveMAS and the corresponding ground-truth across 500 questions. Increasing recursion rounds progressively aligns the generated distribution of RecursiveMAS with the ground truth distribution.

largely aligned by $r = 3$. This aligning trend suggests that RecursiveMAS iteratively refines the latent embeddings and corresponding answers through recursion. We further take a closer look to examine individual test instances and provide detailed case studies in Appendix H. Our case studies reveal a common pattern in which RecursiveMAS may produce an incorrect answer at an early stage, while deeper recursion successfully corrects it through iterative refinement. Together, these analyses provide further evidence that latent thoughts capture semantically meaningful representations, and that deeper recursion improves alignment toward correct final outputs.

Due to page limits, we provide additional analyses of (i) RecursiveLink design and (ii) optimal latent-step ablations, together with (iii) detailed case studies, in Appendices E and H.

7 Related Work

Scaling Reasoning via Recursion. Recent studies explore recursion as an alternative scaling axis for LLMs [48, 49, 50, 51], where the same computation blocks are reused through multiple recurrent rounds (i.e., loops) to increase reasoning depth and iteratively refine hidden representations. One line of work studies recursive language models that apply shared layers to scale latent reasoning. For instance, LoopLM [22] introduces pre-trained looped language models with iterative latent computation. Besides, other work explores other recursive architectures [21, 20, 52], including tiny recursive networks and recursive self-calling schemes for long-context inference. While existing methods in agentic AI primarily focus on recursion inside a single language model, RecursiveMAS exhibits the first attempt to extend the recursive scaling paradigm to system-level.

LLM-based Multi-Agent Systems. Multi-agent systems extend the single LLM paradigm to a collaborative setting [7, 53, 4, 54, 55] by organizing a set of agents with distinct roles that jointly address the problem. A standard multi-agent system topology involves a sequential configuration [56, 9], where agents are assigned in a linear pipeline to decompose and resolve problems in order. Beyond sequential settings, other works also explore mixture-style settings [57, 10, 6], where multiple agents with domain expertise reason in parallel, and their outputs are aggregated into a final decision. Another line of work seeks to improve MAS through textual feedback signals [58, 59, 60, 61]. For example, related optimization methods [16, 43, 13] leverage an LLM to generate natural language feedback to refine contextual inputs and instructions of each agent. Rather than separate training each individual agent or only leveraging textual feedback, RecursiveMAS treats MAS as a unified whole, and scales the system performance via recursively refining the latent information flow.

8 Conclusion

We introduce RecursiveMAS, a recursive multi-agent framework that scales agent collaboration through system-level recursion. Through the RecursiveLink design, RecursiveMAS enables latent thoughts generation within each agent and latent-space interactions across heterogeneous agents. Correspondingly, the inner-outer loop training optimizes the entire system as one integration. Theoretically, our framework leads to more stable training dynamics and improves efficiency compared to text-mediated recursion. Empirical evaluations across mathematical and scientific reasoning, code generation, and search benchmarks demonstrate that RecursiveMAS consistently improves accuracy while greatly reducing inference time and token usage. Overall, RecursiveMAS provides a scalable and efficient framework for multi-agent systems to collaborate and refine iteratively in latent space.

Acknowledgements

The authors thank members of the Zou Group and the iDEA-iSAIL Lab for helpful discussions. This work is supported in part by the National Science Foundation under Award Number 2433308.

References

[1] Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity. *SuperIntelligence-Robotics-Safety & Alignment*, 2(6), 2025.

[2] Peiyang Song, Pengrui Han, and Noah Goodman. Large language model reasoning failures. *arXiv preprint arXiv:2602.06176*, 2026.

[3] 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.

[4] Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*, 2025.

[5] Baixuan Xu, Chunyang Li, Weiqi Wang, Wei Fan, Tianshi Zheng, Haochen Shi, Tao Fan, Yangqiu Song, and Qiang Yang. Towards multi-agent reasoning systems for collaborative expertise delegation: An exploratory design study. *arXiv preprint arXiv:2505.07313*, 2025.

[6] Junlin Wang, WANG Jue, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations*, 2025.

[7] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*, 2024.

[8] Wenyuan Gu, Jiale Han, Haowen Wang, Xiang Li, and Bo Cheng. Explain-analyze-generate: A sequential multi-agent collaboration method for complex reasoning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 7127–7140, 2025.

[9] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 15174–15186, 2024.

[10] Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. X-mas: Towards building multi-agent systems with heterogeneous llms. *arXiv preprint arXiv:2505.16997*, 2025.

[11] Chen Qian, Zihao Xie, YiFei Wang, Wei Liu, Kunlun Zhu, Hanchen Xia, Yufan Dang, Zhuoyun Du, Weize Chen, Cheng Yang, et al. Scaling large language model-based multi-agent collaboration. In *The Thirteenth International Conference on Learning Representations*, 2025.

[12] Harisankar Babu, Philipp Schillinger, and Tamim Asfour. Adaptive domain modeling with language models: A multi-agent approach to task planning. In *2025 IEEE 21st International Conference on Automation Science and Engineering (CASE)*, pages 1701–1708. IEEE, 2025.

[13] Ming Shen, Raphael Shu, Anurag Pratik, James Gung, Yubin Ge, Monica Sunkara, and Yi Zhang. Optimizing llm-based multi-agent system with textual feedback: A case study on software development. *arXiv preprint arXiv:2505.16086*, 2025.

[14] Han Zhou, Xingchen Wan, Ruoxi Sun, Hamid Palangi, Shariq Iqbal, Ivan Vulić, Anna Korhonen, and Sercan Ö Arık. Multi-agent design: Optimizing agents with better prompts and topologies. *arXiv preprint arXiv:2502.02533*, 2025.

[15] Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. *arXiv preprint arXiv:2510.04618*, 2025.

[16] Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip HS Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. Malt: Improving reasoning with multi-agent llm training. *arXiv preprint arXiv:2412.01928*, 2024.

[17] Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*, 2025.

[18] Wanjia Zhao, Mert Yuksekgonul, Shirley Wu, and James Zou. Sirius: Self-improving multi-agent systems via bootstrapped reasoning. *arXiv preprint arXiv:2502.04780*, 2025.

[19] Mengkang Hu, Yuhang Zhou, Wendong Fan, Yuzhou Nie, Bowei Xia, Tao Sun, Ziyu Ye, Zhaoxuan Jin, Yingru Li, Qiguang Chen, et al. Owl: Optimized workforce learning for general multi-agent assistance in real-world task automation. *arXiv preprint arXiv:2505.23885*, 2025.

[20] Alex L Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025.

[21] Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

[22] Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

[23] Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026.

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

[25] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, et al. Gemma 3 technical report, 2025.

[26] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[27] Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickael Rouvier, and Richard Dufour. Biomistral: A collection of open-source pretrained large language models for medical domains. In *Findings of the association for computational linguistics: acl 2024*, pages 5848–5864, 2024.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[29] Jiaru Zou, Ruizhong Qiu, Gaotang Li, Xiyuan Yang, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems. In *Forty-third International Conference on Machine Learning*, 2026.

[30] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[31] HuggingFaceH4. Math-500 dataset. `https://huggingface.co/datasets/HuggingFaceH4/MATH-500`, 2023.

[32] math ai. AIME 2025 dataset. https://huggingface.co/datasets/math-ai/aime25, 2025.

[33] MathArena. Aime 2026 dataset. https://huggingface.co/datasets/MathArena/aime_2026, 2026.

[34] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.

[35] Hang Yang, Hao Chen, Hui Guo, Yineng Chen, Ching-Sheng Lin, Shu Hu, Jinrong Hu, Xi Wu, and Xin Wang. Llm-medqa: Enhancing medical question answering through case studies in large language models. *arXiv preprint arXiv:2501.05464*, 2024.

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

[37] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572, 2023.

[38] 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, 2018.

[39] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711, 2023.

[40] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report, 2025.

[41] 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.

[42] John Schulman and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. https://thinkingmachines.ai/blog/lora/.

[43] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

[44] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20275–20321. Association for Computational Linguistics, November 2025.

[45] Xiaoke Huang, Juncheng Wu, Hui Liu, Xianfeng Tang, and Yuyin Zhou. m1: Unleash the potential of test-time scaling for medical reasoning with large language models, 2025.

[46] Wasi Uddin Ahmad, Sean Narenthiran, Somshubra Majumdar, Aleksander Ficek, Siddhartha Jain, Jocelyn Huang, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning: Advancing data distillation for competitive coding, 2025.

[47] Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*, 2025.

[48] Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *arXiv preprint arXiv:2502.05171*, 2025.

[49] Yinxi Li, Jiaao Chen, Fang Wu, Jiakai Yu, Heli Qi, Weihao Xuan, Haokai Zhao, Pengyu Nie, Di Jin, and Xiangru Tang. Learning multi-step reasoning via persistent latent state propagation. In *Workshop on Latent {\&} Implicit Thinking {\textendash} Going Beyond CoT Reasoning*, 2026.

[50] Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, et al. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. *arXiv preprint arXiv:2507.10524*, 2025.

[51] Guo Tang, Shixin Jiang, Heng Chang, Nuo Chen, Yuhan Li, Huiming Fan, Jia Li, Ming Liu, and Bing Qin. Looprpt: Reinforcement pre-training for looped language models. *arXiv preprint arXiv:2603.19714*, 2026.

[52] Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025.

[53] Yingxuan Yang, Qiuying Peng, Jun Wang, Ying Wen, and Weinan Zhang. Llm-based multi-agent systems: Techniques and business perspectives. *arXiv preprint arXiv:2411.14033*, 2024.

[54] Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, Yi Dong, Evelina Bakhturina, Tao Yu, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Toolorchestra: Elevating intelligence via efficient model and tool orchestration, 2025.

[55] Sean Wu, Aneesh Pappu, Jiaru Zou, Yubin Kim, Yuncong Mao, Hongjian Zhou, Junchi Yu, Jieneng Chen, and James Zou. Measuring ai agents as teammates with collaborative intelligence, 2026.

[56] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: communicative agents for "mind" exploration of large language model society. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[57] Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Zou, Guohao Li, and Tianlong Chen. Graph-of-agents: A graph-based framework for multi-agent llm collaboration. In *The Fourteenth International Conference on Learning Representations*, 2026.

[58] Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[59] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987, 2023.

[60] Monishwaran Maheswaran, Leon Lakhani, Zhongzhu Zhou, Shijia Yang, Junxiong Wang, Coleman Hooper, Yuezhou Hu, Rishabh Tiwari, Jue Wang, Harman Singh, et al. Squeeze evolve: Unified multi-model orchestration for verifier-free evolution. *arXiv preprint arXiv:2604.07725*, 2026.

[61] Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. Benchmarking large language models on answering and explaining challenging medical questions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3563–3599, 2025.

[62] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

[63] Tavily. Tavily search api. `https://www.tavily.com`, 2026.

[64] Hugging Face. Transformers documentation. https://huggingface.co/docs/transformers/en/index, 2025.

[65] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pages 611–626, 2023.

[66] Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. *arXiv preprint arXiv:2510.05592*, 2025.

[67] Xinlei Yu, Zhangquan Chen, Yongbo He, Tianyu Fu, Cheng Yang, Chengming Xu, Yue Ma, Xiaobin Hu, Zhe Cao, Jie Xu, et al. The latent space: Foundation, evolution, mechanism, ability, and outlook. *arXiv preprint arXiv:2604.02029*, 2026.

[68] Zhuoyun Du, Runze Wang, Huiyu Bai, Zouying Cao, Xiaoyong Zhu, Yu Cheng, Bo Zheng, Wei Chen, and Haochao Ying. Enabling agents to communicate entirely in latent space. *arXiv preprint arXiv:2511.09149*, 2025.

[69] Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. *arXiv preprint arXiv:2510.03215*, 2025.

[70] Hancheng Ye, Zhengqi Gao, Mingyuan Ma, Qinsi Wang, Yuzhe Fu, Ming-Yu Chung, Yueqian Lin, Zhijian Liu, Jianyi Zhang, Danyang Zhuo, et al. Kvcomm: Online cross-context kv-cache communication for efficient llm-based multi-agent systems. *arXiv preprint arXiv:2510.12872*, 2025.

[71] Yujia Zheng, Zhuokai Zhao, Zijian Li, Yaqi Xie, Mingze Gao, Lizhu Zhang, and Kun Zhang. Thought communication in multiagent collaboration. *arXiv preprint arXiv:2510.20733*, 2025.

[72] Fengkai Yang, Zherui Chen, Xiaohan Wang, Xiaodong Lu, Jiajun Chai, Guojun Yin, Wei Lin, Shuai Ma, Fuzhen Zhuang, Deqing Wang, Yaodong Yang, Jianxin Li, and Yikun Ban. Your group-relative advantage is biased. *arXiv preprint arXiv:2601.08521*, 2026.

[73] Zhixia Zhang, Zixuan Huang, Gongxun Li, Huaiyang Wang, Chengyi Yuan, Xin Xia, Deqing Wang, Fuzhen Zhuang, Shuai Ma, Ning Ding, Yaodong Yang, Jianxin Li, and Yikun Ban. Heterogeneous agent collaborative reinforcement learning. *arXiv preprint arXiv:2603.02604*, 2026.

[74] Yadong Lu, Chunyuan Li, Haotian Liu, Jianwei Yang, Jianfeng Gao, and Yelong Shen. An empirical study of scaling instruct-tuned large multimodal models. *arXiv preprint arXiv:2309.09958*, 2023.

[75] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Table of Contents

A Theoretical Analysis **17**

A.1 Running Complexity Analysis 17

A.2 Learning Advantage Analysis 17

B Experiment Setups **19**

B.1 Evaluation Datasets 19

B.2 Model Configurations of RecursiveMAS 19

B.3 Compared Baselines 19

B.4 Additional Implementation Details 20

C Additional Related Work **22**

D Additional Experiments **22**

D.1 Results on Different Collaboration Patterns 22

D.2 Training Cost Analysis 23

E In-depth Analyses on RecursiveMAS **23**

E.1 Effectiveness of RecursiveLink Design 23

E.2 Optimal Length of Latent Thoughts Generation 23

F Broader Impact and Limitations **24**

G Examples of RecursiveMAS Across Different Downstream Tasks **25**

H Case Study on RecursiveMAS across Different Recursion Rounds **29**

I Prompt Template for RecursiveMAS **32**

Appendix

A Theoretical Analysis

A.1 Running Complexity Analysis

Proposition A.1 (Restate of Proposition 3.1). *Without RecursiveLink, a text-based Recursive MAS with the same collaboration structure requires runtime complexity of* $\Theta(N(m|V|d_h + (t+m)d_h^2 + (t+m)^2 d_h))$*; In contrast, with RecursiveLink-enabled collaboration, RecursiveMAS achieves an end-to-end runtime complexity of* $\Theta(N(md_h^2 + (t+m)d_h^2 + (t+m)^2 d_h))$.

Proof of Proposition 3.1. We analyze the runtime complexity for each agent and then extend the result to the full multi-agent system. For each single agent, given the context length is at most t, and the generation length is at most m, the Transformer processes a sequence of length at most $t+m$, requiring $\Theta((t+m)d_h^2)$ time for the feed-forward layers and $\Theta((t+m)^2 d_h)$ time for self-attention. This standard Transformer computation is shared by both RecursiveMAS and text-based Recursive MAS.

The computational difference comes from how RecursiveMAS process the generated embeddings. In RecursiveMAS, each of the m latent embeddings is transformed by RecursiveLink, which incurs an additional cost of $\Theta(md_h^2)$. In text-based Recursive MAS, each embedding must be decoded into an explicit token by projecting it to the vocabulary space and computing logits over $|V|$ tokens, resulting in an additional cost of $\Theta(m|V|d_h)$.

Adding all terms together, our proposed RecursiveMAS needs $\Theta(md_h^2+(t+m)d_h^2+(t+m)^2 d_h)$ time for each agent, while text-based Recursive MAS needs $\Theta(m|V|d_h + (t+m)d_h^2 + (t+m)^2 d_h)$ time. Since there are N agents in the system, our proposed RecursiveMAS needs $\Theta(N(md_h^2 + (t+m)d_h^2 + (t+m)^2 d_h))$ time, and text-based Recursive MAS needs $\Theta(N(m|V|d_h + (t+m)d_h^2 + (t+m)^2 d_h))$ time in total. □

A.2 Learning Advantage Analysis

Assumption A.1. *Text-based SFT can be regarded as using* $\mathcal{R}_{text}(h) = W_{in}\,\mathrm{softmax}(W_{out}h)$ *as the recursive link, where* W_{in} *is the token-to-embedding matrix, and* W_{out} *denotes the embedding-to-logits matrix. We also assume* $\|W_{in}\|_2 \le O(1)$ *and* $\|W_{out}\|_2 \le O(1)$*. For RecursiveLink* $\mathcal{R}(h)$*, we assume that* W_1, W_2 *follow Kaiming normal initialization, and we only analyze the case where* $W_3 = I$.

Theorem A.2 (Restate of Theorem 4.1). *Under the Realistic Assumptions (stated in Assumption A.1), if tokens are confident with entropy* $\le \epsilon$*, where typically* $\epsilon \ll 1$*: directly applying text-based SFT (denoted by* $\mathcal{R}_{text}(h)$*) during recursion suffers from gradient vanishing (i.e., gradient norm close to 0); while RecursiveMAS with the RecursiveLink* $\mathcal{R}$ *maintains stable and near constant gradients (i.e., gradient norm close to 1) during looped backpropagation process. Formally, with probability* $\ge 1-\delta$*, we have*

$$\left\| \frac{\partial \mathcal{R}_{text}(h)}{\partial h} \right\|_2 \le O(\epsilon) \ll 1, \qquad \left\| \frac{\partial \mathcal{R}(h)}{\partial h} \right\|_2 \ge \Omega\left(1 - \sqrt{\frac{1}{d_h}\log\frac{1}{\delta}}\right). \tag{8}$$

Proof of Theorem 4.1. We first analyze the gradient behavior of text-based recursive interaction. By applying the chain rule to $\mathcal{R}_{\text{text}}(h)$, the gradient matrix is:

$$J_{\text{text}} = \frac{\partial \mathcal{R}_{\text{text}}(h)}{\partial h} = W_{\text{in}} S W_{\text{out}}, \qquad S = \mathrm{diag}(p) - pp^T,$$

where $p = \mathrm{softmax}(W_{\text{out}}h)$ is the next token distribution. Then, by the sub-multiplicativity of the spectral norm,

$$\|J_{\text{text}}\|_2 \le \|W_{\text{in}}\|_2 \|S\|_2 \|W_{\text{out}}\|_2 \le O(1) \cdot \|S\|_2 \cdot O(1) = O(\|S\|_2).$$

Because S represents the covariance matrix of a categorical distribution, it is symmetric and positive semi-definite. Thus, its spectral norm is upper-bounded by its trace:

$$\|S\|_2 \leq \text{Tr}(S) = \sum_{i=1}^{|V|}(p_i - p_i^2) = \sum_{i=1}^{|V|} p_i - \sum_{i=1}^{|V|} p_i^2 = 1 - \|p\|_2^2.$$

Using the logarithm inequality $\ln z \leq z - 1$ (for all $z > 0$), we can lower-bound the entropy:

$$\epsilon \geq \sum_{i=1}^{|V|} p_i(-\ln p_i) \geq \sum_{i=1}^{|V|} p_i(1 - p_i) = 1 - \|p\|_2^2.$$

Substituting this inequality back into the norm bound yields:

$$\|S\|_2 \leq 1 - \|p\|_2^2 \leq \epsilon.$$

Therefore, combining the constants, we have:

$$\left\| \frac{\partial \mathcal{R}_{\text{text}}(h)}{\partial h} \right\|_2 = \|J_{\text{text}}\|_2 \leq O(\epsilon).$$

We next analyze the gradient behavior of RecursiveMAS. Applying the chain rule to $\mathcal{R}(h)$, the gradient matrix is:

$$J = \frac{\partial \mathcal{R}(h)}{\partial h} = I + W_2 \Sigma' W_1,$$

where $\Sigma' = \text{diag}(\sigma'(W_1 h))$. By the triangle inequality for the matrix norm,

$$\Big| \|J\|_2 - 1 \Big| = \Big| \|J\|_2 - \|I\|_2 \Big| \leq \|J - I\|_2 = \|W_2 \Sigma' W_1\|_2.$$

Since W_1, W_2 follow Kaiming initialization, and the GELU function σ has $|\sigma'| \leq O(1)$, then by the subgaussian matrix concentration inequality, $\|W_2 \Sigma' W_1\|_2 \leq O\left(\sqrt{\frac{1}{d_h} \log \frac{1}{\delta}} + 1\right)$ with probability $\geq 1 - \delta$. It follows that:

$$\|J\|_2 \geq \Omega\left(1 - \sqrt{\frac{1}{d_h} \log \frac{1}{\delta}}\right).$$

□

B Experiment Setups

B.1 Evaluation Datasets

We introduce all datasets used in our experiments as follows:

Mathematical Reasoning.

- **MATH500** [31] is a widely used subset of the MATH benchmark, containing mathematical problems across algebra, geometry, number theory, probability, and combinatorics.
- **AIME2025** [32] contains 30 challenging problems from the 2025 American Invitational Mathematics Examination. These problems require olympiad-style derivations and precise numerical answers, providing a compact but difficult benchmark for mathematical reasoning.
- **AIME2026** [33] follows the same AIME-style questions with 30 challenging competition-level math problems. We use it to further test performance and generalization on difficult mathematical reasoning tasks.

Scientific and Medical Tasks.

- **GPQA-Diamond** [34] is the most difficult split of GPQA, consisting of graduate-level multiple-choice questions in biology, physics, and chemistry. It requires specialized scientific knowledge and careful multi-step reasoning beyond shallow factual recall.
- **MedQA** [35] contains medical licensing-style questions that assess biomedical knowledge, clinical reasoning, and diagnostic decision-making. The benchmark requires integrating domain-specific evidence from patient scenarios and selecting the most appropriate answer.

Code Generation. ("Code Gen.")

- **LiveCodeBench-v6** [36] is a contamination-resistant code generation benchmark built from recent programming problems. It evaluates whether models can synthesize functionally correct programs under realistic problem specifications and hidden test cases.
- **MBPP Plus** [37] extends the original MBPP benchmark with more comprehensive test cases for Python program synthesis. The stricter execution-based evaluation provides a more reliable measure of functional correctness.

For code generation tasks, unless otherwise specified, we evaluate the RecursiveMAS Light and Scaled settings on MBPP+ and LiveCodeBench, respectively.

Search-based Question Answering.

- **HotpotQA** [38] is a multi-hop question answering benchmark based on Wikipedia. It requires models to gather and combine evidence from multiple supporting facts, making it suitable for evaluating search-based reasoning.
- **Bamboogle** [39] is a compact but challenging benchmark for search-intensive multi-hop reasoning. Its questions often require decomposition and intermediate retrieval steps before composing the final answer.

B.2 Model Configurations of RecursiveMAS

Detailed model configurations for RecursiveMAS are provided in Table 3. We select off-the-shelf models from diverse families to form heterogeneous agent compositions with complementary strengths. Each model is assigned to match the role-specific demands of its collaboration pattern while preserving practical efficiency and scalability.

B.3 Compared Baselines

We compare our method against the following baselines:

Single-Agent Fine-tuning Baselines.

Table 3: **Model configurations for different collaboration patterns in RecursiveMAS.** We select off-the-shelf models from diverse model families to form heterogeneous agent compositions with complementary strengths. Each assignment is chosen to match the role-specific needs of the corresponding collaboration pattern while preserving both practical efficiency and scalability.

Collaboration Pattern	Role	Model Size & Version
Sequential Style (Light)	Planner	Qwen3-1.7B [41]
	Critic	Llama3.2-1B-Instruct [24]
	Solver	Qwen2.5-Math-1.5B-Instruct [40]
Sequential Style (Scaled)	Planner	Gemma3-4B-it [25]
	Critic	Llama3.2-3B-Instruct [24]
	Solver	Qwen3.5-4B [41]
Mixture Style	Code Specialist	Qwen2.5-Coder-3B-Instruct [62]
	Science Specialist	BioMistral-7B [27]
	Math Specialist	DeepSeek-R1-Distill-Qwen-1.5B [40]
	Summarizer	Qwen3.5-2B [41]
Distillation Style	Learner	Qwen3.5-4B [41]
	Expert	Qwen3.5-9B [41]
Deliberation Style	Reflector	Qwen3.5-4B [41]
	Tool-Caller	Qwen3.5-4B (with Tool-Integration) [41]

- **Single Agent (w/ LoRA)** uses the final agent from the corresponding collaboration pattern and trains it with LoRA adapters using the same training data as RecursiveMAS.
- **Single Agent (w/ Full-SFT)** further fine-tunes all parameters of the same single-agent backbone using fully supervised fine-tuning.

Representative Multi-Agent Frameworks.

- **Mixture-of-Agents (MoA)** [6] organizes multiple LLM agents into a layered multi-agent system, where agents in each layer aggregate responses from the previous layer to produce refined outputs as the final answer.
- **TextGrad** [43] optimizes multi-agent systems by propagating natural-language feedback as textual gradients. We use TextGrad as a baseline method for text-mediated system optimization.

Recursion-based Methods.

- **LoopLM** [22] is a looped language model that performs recursive computation with shared transformations in latent space. In our evaluation, we mainly adopt the Ouro-2.6B model.
- **Recursive-TextMAS** uses the same recursive multi-agent collaboration structure as RecursiveMAS, but agents communicate through explicit text rather than latent representations.

B.4 Additional Implementation Details

Training Data Curation. To optimize RecursiveMAS under our inner-outer training pipeline, we construct role-specific supervision targets for each agent across all collaboration patterns. We start by collecting question-answer pairs as raw training samples from four domains, including s1K [44], m1K [45], OpenCodeReasoning [46], and ARPO-SFT [47]. For each training sample, we rewrite the original answer into agent-level targets according to the role assignments of each collaboration pattern, as follows:

- For *Sequential-Style*, we use Qwen3.5-397B-A17B to rewrite the answers into an initial step-by-step plan and a refined critic-guided plan. During training, the initial plan is used as the supervision

target for the Planner, the critic-guided plan is used for the Critic, and the original answer is retained for the Solver.

- For *Mixture-Style*, each specialist in the MAS first generates responses for questions from its specialized domain, and these responses are then used to supervise the corresponding specialist. The ground truth answers are used as targets for the Summarizer.
- For *Distillation-Style*, the Expert first generates guidance-style responses for each training sample, which are then used as supervision targets for the Expert. The Learner is supervised directly by the ground-truth answers.
- For *Deliberation-Style*, we use the ground truth answers as the supervision targets for both the Reflector and Tool-Caller agent.

After the role-specific data construction, each agent is assigned its own training pairs, where the input consists of the original question, and the output is the corresponding role-based response. These agent-level pairs are then used as the supervision data for subsequent training.

Implementation Details. During training, all base LLMs' parameters are frozen, and we only update the inner and outer RecursiveLink using AdamW with a batch size of 4 and a maximum sequence length of 4096 tokens. The maximum output length is adjusted for each task based on its relative difficulty. We perform hyperparameter tuning and report the mean performance over five independent runs. During inference, the maximum generation length is set for 2000 tokens for MATH500, 4000 tokens for MedQA, GPQA-Diamond, LiveCodeBench, and MBPP Plus, and 16000 tokens for AIME2025/2026. For all Qwen models, we enables the official Instruct mode [23] for more efficient and controllable answer generation. For the Deliberation-Style MAS, we provide a standard Python environment and a Tavily [63] search API as external tools. We implement RecursiveMAS and baselines with both HuggingFace Transformer [64] and vLLM backend [65]. All experiments are conducted on H100 and A100 GPUs.

Evaluation Protocol. Across all non-code generation tasks, we first normalize the extracted answer by removing extra whitespace, stripping punctuation, and converting text to lowercase before applying task-specific correctness checks.

- For *numerical problems (MATH500, AIME2025, AIME2026)*, we compare the numerical value of the extracted answer with the ground truth. The answer is considered correct if the two values are mathematically equivalent.
- For *multiple-choice questions (GPQA-Diamond MedQA)*, we directly compare the predicted choice with the ground truth letter, where an exact option match is counted as correct.
- For *code generation tasks (LiveCodeBench and MBPP Plus)*, we first extract the generated code block and then execute it with provided test cases in a sandboxed python environment. Each individual test case is assigned a timeout of 10 seconds to prevent non-terminating programs.
- For *search-based tasks (HotpotQA, Bamboogle)*, we follow the standard LLM-as-a-judge evaluation method [66] and use the Qwen3.5-397B-A17B model as a binary judge to determine whether the generated answer is correct with respect to the ground truth answer.

When an output reaches the maximum generation length without producing an extractable answer, we follow standard early-stopping evaluation methods [44, 41] by appending "Final Answer:" to the model output to elicit a final response.

Table 4: Comparison of RecursiveMAS in Distillation-Style Multi-agent System. RecursiveMAS improves the Learner agent by 8.0% via distilling knowledge from the Expert agent while retaining a 1.5× end-to-end speed advantage over the Expert agent.

Method (Distillation-Style)	Metric	AIME2026	GPQA-D	LiveCodeBench	MBPP+	MedQA
Expert Model	Acc.	90.0	72.7	46.2	73.4	86.0
	Time	9473	2558	9352	2342	2124
Learner Model	Acc.	76.7	61.4	38.4	67.5	77.9
	Time	4495	1289	5396	1171	1183
RecursiveMAS	Acc.	83.3	70.0	40.1	71.9	83.0
	Time	5967	1671	6863	1516	1436

C Additional Related Work

Latent-Space Collaboration. Beyond text-based interaction, recent studies have explored leveraging the latent space as an alternative medium for LLM communication. One line of work studies transferring hidden embeddings for cross-model communication [67, 68], while other works investigate reusing internal states to share information across LLMs [69, 70]. Recent studies extend this scheme to agentic settings, where latent interfaces are used to support coordination among multiple agents [71, 29]. Different from these studies, RecursiveMAS treats latent information as part of a system-level recursive information flow, enabling heterogeneous agents to recursively collaborate and improve as a unified MAS.

Collaborative Agent Training. Beyond inference-time collaboration, recent studies investigate how to optimize agents from verifiable rewards. Yang et al. [72] identify a systematic difficulty-dependent bias in group-relative advantage estimation and propose a history-aware reweighting method to mitigate this bias, providing a principled approach to improving group-based optimization. Zhang et al. [73] introduce a collaborative training framework that enables heterogeneous agents to mutually improve through verified trajectory sharing while preserving independent execution at inference time. Our inner-outer loop learning algorithm is orthogonal to these works: it optimizes the entire recursive system through shared gradient-based credit assignment across recursion rounds, and could be combined with improved advantage estimators or cross-agent trajectory sharing in future work.

D Additional Experiments

D.1 Results on Different Collaboration Patterns

Table 5: Comparison of RecursiveMAS in Mixture-Style Multi-agent System.

Method (Mixture-Style)	AIME2026	GPQA-Diamond	LiveCodeBench	MedQA
Math Specialist	43.3	37.4	18.9	29.0
Code Specialist	13.3	26.2	21.5	43.3
Science Specialist	10.0	27.0	7.6	48.1
RecursiveMAS	**46.7**	**43.0**	**23.8**	**61.7**

Table 6: Comparison of RecursiveMAS in Deliberation-Style Multi-agent System.

Method (Deliberation-Style)	AIME2026	GPQA-Diamond	HotpotQA	Bamboogle
Reflector	76.7	61.2	27.5	40.9
Tool-Caller	86.7	63.1	39.6	49.8
RecursiveMAS	90.0	65.0	41.4	53.7

We report the detailed results of RecursiveMAS under three additional collaboration patterns in Tables 4, 5, and 6, corresponding to the summarized results in Figure 5.

In *Mixture-style*, RecursiveMAS achieves an average improvement of 6.2% over the strongest domain specialist on each benchmark, suggesting that recursive interaction enables non-trivial cross-domain composition beyond what can be attained by selecting one individual specialist alone. In *Deliberation-style*, we evaluate tool use on both mathematical and search-intensive tasks. RecursiveMAS improves the original tool-calling agent by 6.5%, showing that recursive latent coordination remains effective in tool-calling settings through iterative interaction with the Reflector. Finally, in *Distillation-style*, RecursiveMAS improves the learner by 8.0% while retaining 1.5× end-to-end speed advantage over the expert. In this way, RecursiveMAS distills much of the expert's capability into a more efficient system. Overall, these results show that RecursiveMAS provides both performance gains and efficiency benefits across diverse MAS collaboration patterns.

Table 7: **Cost analysis on RecursiveMAS.** We report the peak GPU memory usage (GB), number of trainable parameters, estimated cost, and average accuracy (%) across all downstream tasks.

Methods	GPU Mem.	Trainable Param.	Cost	Avg. Acc.
LoRA Training	21.67	15.92M (0.37%)	$6.64	66.9
Full-SFT	41.40	4.21B (100%)	$9.67	68.6
RecursiveMAS	**15.29**	**13.12M (0.31%)**	**$4.27**	**74.9**

D.2 Training Cost Analysis

We further analyze the training cost of RecursiveMAS under the scaled sequential-style MAS setting. We compare RecursiveMAS with direct training methods, including LoRA and full supervised fine-tuning with the same training data and backbone setup. For cost estimation, we follow prior methods [74, 75] to measure the cost based on GPU usage. As shown in Table 7, RecursiveMAS utilizes the lowest per-agent GPU memory, trainable parameter count, and estimated cost among all compared training strategies. Meanwhile, RecursiveMAS achieves the highest accuracy across all downstream tasks, suggesting that optimizing the lightweight RecursiveLink provides a better cost-performance trade-off than other training methods.

E In-depth Analyses on RecursiveMAS

E.1 Effectiveness of RecursiveLink Design

To validate the effectiveness of RecursiveLink, we compare our 2-layer residual design against three alternatives: (i) a 1-layer network, (ii) a 1-layer network with the residual connection, and (iii) a 2-layer network without the residual connection. We conduct experiments using the scaled sequential-style RecursiveMAS and adapt the same architecture for $\mathcal{R}_{\text{in}}$ and $\mathcal{R}_{\text{out}}$.

Table 8: **Efficacy on RecursiveLink Design.** We compare accuracy across alternative architectural designs.

RecursiveLink Design	Math500	GPQA-D	LiveCodeBench
1-Layer	84.4	63.2	40.1
Res+1-Layer	86.7	65.3	41.4
2-Layer	85.6	64.5	40.5
Res+2-Layer (ours)	**88.0**	**66.2**	**42.9**

As shown in Table 8, our 2-layer residual design performs best across all three benchmarks, and the residual connection delivers additional improvements across different backbone models. For example, on GPQA-Diamond, equipping a single-layer design with a residual branch improves the performance from 63.2% to 65.3%, which is even higher than the plain 2-layer design (64.5%). These results align with our design intuition in Section 3.1: by preserving latent semantics while learning only the distributional shift, RecursiveLink achieves stable training and stronger inference performance.

E.2 Optimal Length of Latent Thoughts Generation

We next study and ablate the latent thoughts length m to examine how much of each agent's internal reasoning is sufficient to support effective collaboration. Under the scaled sequential-style of RecursiveMAS, we evaluate a broad range of m. As illustrated in Figure 8, increasing m improves performance in the early regime. Once m reaches a moderate scale (around $m = 80$), performance is stabilized across all benchmarks. The ablation suggests that RecursiveMAS enables effective agent reasoning and interaction with only a modest latent-thought budget, in sharp contrast to text-based collaboration that typically requires longer CoT and costly token generation.

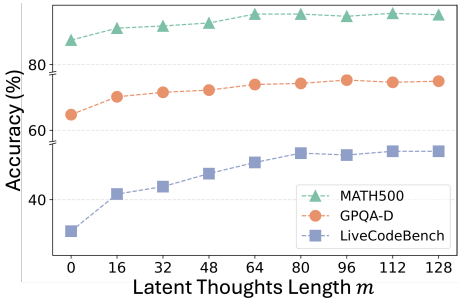

Figure 8: Effectiveness of RecursiveMAS's latent thoughts with different step lengths.

Table 9: Ablation Study on Length of Latent Thoughts m transferred across agents on RecursiveMAS.

Latent Steps	**0**	**16**	**32**	**48**	**64**	**80**	**96**	**112**	**128**
Math500	83.3	84.9	85.2	85.6	86.8	86.8	86.5	86.9	86.7
GPQA-D	61.4	62.0	62.8	63.6	64.1	64.2	64.5	64.3	64.4
LiveCodeBench	38.1	40.3	40.7	41.4	42.0	42.5	42.2	42.6	42.6

We further provide detailed ablation results on the length of latent thoughts m in Table 9, corresponding to the plot in Figure 8. As m increases, RecursiveMAS consistently improves across all benchmarks, and the performance gradually saturates around $m = 80$, suggesting that a moderate latent thought budget is sufficient for effective latent collaboration.

F Broader Impact and Limitations

Broader Impact. RecursiveMAS improves the reliability and efficiency of multi-agent systems by enabling agents to collaborate through latent-space recursion with lightweight trainable links. By reducing repeated intermediate text generation, our framework lowers inference cost and token usage while preserving strong task performance across diverse domains. The method is general-purpose and can be applied to standard reasoning, search, medical, and code-generation benchmarks. While our method has broad applicability across domains, we do not foresee any specific societal risks or negative impacts that require special consideration. In addition, we ensure all adopted language models are deployed responsibly and follow standard instructions and protocols.

Limitations. While RecursiveMAS shows consistent gains across diverse tasks and collaboration patterns, several directions remain open for future work. Our current experiments focus on representative MAS structures, including sequential, mixture, distillation, and deliberation styles. While these settings cover most common collaboration patterns, extending RecursiveMAS to additional agent topologies and societies is an interesting future direction. Overall, RecursiveMAS provides a promising step toward efficient and scalable multi-agent collaboration. We hope future work could build on this framework to develop more adaptive and broadly applicable multi-agent systems.

G Examples of RecursiveMAS Across Different Downstream Tasks

Case Study of RecursiveMAS on Mathematical Reasoning Questions (AIME2026)

Question:
A plane contains points A and B with $AB = 1$. Point A is rotated in the plane counterclockwise through an acute angle θ around point B to point A'. Then B is rotated in the plane clockwise through angle θ around point A' to point B'. Suppose that $AB' = \frac{4}{3}$. The value of $\cos\theta$ can be written as $\frac{m}{n}$, where m and n are relatively prime positive integers. Find $m + n$.

Recursion Round 1 (in latent space):
Planner $\rightarrow$ ***Critic*** $\rightarrow$ ***Solver*** $\rightarrow$

Recursion Round 2 (in latent space):
Planner $\rightarrow$ ***Critic*** $\rightarrow$ ***Solver*** $\rightarrow$

Recursion Round 3 (with final output):
Planner $\rightarrow$ ***Critic*** $\rightarrow$ ***Solver*:**
To solve the problem, we need to analyze the geometric transformations and use the given distances to find the value of $\cos\theta$.

1. **Initial Setup:** Points A and B are in the plane with $AB = 1$. Point A is rotated counterclockwise through an acute angle θ around point B to point A'. Point B is then rotated clockwise through the same angle θ around point A' to point B'. We are given that $AB' = \frac{4}{3}$.
2. **Coordinate System:** Place point B at the origin $(0, 0)$ and point A at $(1, 0)$.
3. **Rotation of Point A:** When point A is rotated counterclockwise by angle θ around point B, its new coordinates A' are: $A' = (1\cos\theta, 1\sin\theta) = (\cos\theta, \sin\theta)$.
4. **Rotation of Point B:** Point B is rotated clockwise by angle θ around point A'. The coordinates of B' after this rotation can be found using: $B' = A' + R_{-\theta}(B - A')$, where $R_{-\theta}$ is the rotation matrix for clockwise rotation by θ:

$$R_{-\theta} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}.$$

   The coordinates of $B - A'$ are $B - A' = (0 - \cos\theta, 0 - \sin\theta) = (-\cos\theta, -\sin\theta)$. Applying the rotation matrix:

$$R_{-\theta}(B - A') = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} -\cos\theta \\ -\sin\theta \end{pmatrix} = \begin{pmatrix} -\cos^2\theta - \sin^2\theta \\ \cos\theta\sin\theta - \sin\theta\cos\theta \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}.$$

   Therefore, the coordinates of B' are: $B' = (\cos\theta, \sin\theta) + (-1, 0) = (\cos\theta - 1, \sin\theta)$.
5. **Distance AB':** The distance AB' is given by:

$$AB' = \sqrt{(\cos\theta - 1 - 1)^2 + (\sin\theta - 0)^2} = \sqrt{(\cos\theta - 2)^2 + \sin^2\theta}.$$

   Simplifying inside the square root:

$$AB' = \sqrt{\cos^2\theta - 4\cos\theta + 4 + \sin^2\theta} = \sqrt{1 - 4\cos\theta + 4} = \sqrt{5 - 4\cos\theta}.$$

   We are given that $AB' = \frac{4}{3}$, so $\sqrt{5 - 4\cos\theta} = \frac{4}{3}$. Squaring both sides gives $5 - 4\cos\theta = \frac{16}{9}$. Solving for $\cos\theta$: $5 - \frac{16}{9} = 4\cos\theta \implies \frac{45}{9} - \frac{16}{9} = 4\cos\theta \implies \frac{29}{9} = 4\cos\theta \implies \cos\theta = \frac{29}{36}$.
6. **Final Answer:** The value of $\cos\theta$ is $\frac{29}{36}$. The sum of the numerator and the denominator is $m + n = 29 + 36 = 65$.

Thus, the final answer is $\boxed{65}$. *Correct Answer* ✓

Case Study of RecursiveMAS on Scientific and Medical Questions (AIME2026)

Question:
A 45-year-old mechanic presents to the emergency department complaining of acute-onset shortness of breath while repairing a plowing tractor for his neighbor. The patient denies having any history of asthma or respiratory symptoms, and does not smoke. His temperature is 99.8°F (37.7°C), pulse is 65/min, blood pressure is 126/86 mmHg, and respirations are 20/min. His oxygen saturation is 97%. On exam, he is pale and diaphoretic. His pupils are contracted. Diffuse wheezes are noted in all lung fields. What is the best treatment for his condition?
A. Succinylcholine
B. Inhaled ipratropium and oxygen
C. Atropine and pralidoxime
D. Inhaled albuterol and oxygen

Recursion Round 1 (in latent space):
Planner → ***Critic*** → ***Solver*** →

Recursion Round 2 (in latent space):
Planner → ***Critic*** → ***Solver*** →

Recursion Round 3 (with final output):
Planner → ***Critic*** → ***Solver*:**
The clinical presentation is highly suggestive of organophosphate poisoning. Clinical Signs:

- **Dizziness/Shortness of breath**: Suggests respiratory distress.
- **Pupils are contracted (Miosis)**: This is a classic sign of cholinergic excess (SLUDGE mnemonic: Salivation, Lacrimation, Urination, Defecation, GI upset, Emesis).
- **Diffuse wheezes**: Caused by bronchorrhea and bronchospasm due to overstimulation in the lungs.
- **Pale and diaphoretic**: Diaphoresis is a key sign of cholinergic crisis.
- **Vitals**: The patient is tachypneic (20/min is borderline but in context of distress, this is significant) and hypoxic (implied by distress, though 97% is maintained, the saturation will drop quickly without treatment).

Analysis of Options:

- **A. Succinylcholine**: Depolarizing neuromuscular blocker. Exacerbates or prolongs paralysis due to high endogenous acetylcholine; does not resolve the underlying cholinergic crisis.
- **B. Inhaled ipratropium and oxygen**: Anticholinergic. Relieves local respiratory symptoms but fails to address the root cause and systemic cholinergic toxicity.
- **C. Atropine and pralidoxime**: **Correct treatment**. Atropine (muscarinic antagonist) is life-saving, reversing severe secretions and bronchospasm. Pralidoxime reactivates acetylcholinesterase to resolve nicotinic effects.
- **D. Inhaled albuterol and oxygen**: Beta-2 agonist. May help bronchospasm but ineffective against massive bronchorrhea and systemic poisoning.

Conclusion: The patient is suffering from organophosphate poisoning (cholinergic crisis). The standard of care involves the administration of atropine to reverse muscarinic effects (especially respiratory secretions and bronchospasm) and pralidoxime to regenerate acetylcholinesterase. So the final answer is: $\boxed{C}$ *Correct Answer* ✓

Case Study of RecursiveMAS on Code Generation Tasks (MBPP Plus)

Question:
Write a function to check if each element of second tuple is smaller than its corresponding element in the first tuple.

Recursion Round 1 (in latent space):
Planner → ***Critic*** → ***Solver*** →

Recursion Round 2 (in latent space):
Planner → ***Critic*** → ***Solver*** →

Recursion Round 3 (with final output):
Planner → ***Critic*** → ***Solver*:**
To solve this problem, we need to write a function that takes two tuples as input and checks if each element of the second tuple is smaller than its corresponding element in the first tuple. If all elements satisfy this condition, the function should return True; otherwise, it should return False. Here is the code logic:

1. Define the function ‘check_smaller‘ that takes two tuples as input.
2. Check if the lengths of the two tuples are equal. If they are not, return False.
3. Iterate over the elements of the tuples and check if each element of the second tuple is smaller than its corresponding element in the first tuple.
4. If all elements satisfy the condition, return True; otherwise, return False.

Here is the Python code for the solution:

```
def check_smaller(tuple1, tuple2):
    # Check if the lengths of the tuples are equal
    if len(tuple1) != len(tuple2):
        return False

    # Iterate over the elements of the tuples
    for i in range(len(tuple1)):
        if tuple1[i] <= tuple2[i]:
            return False

    # If all elements satisfy the condition, return True
    return True
```

Correct Answer ✓

Case Study of RecursiveMAS on Search Based Tasks (Bamboogle)

Question:
Who was president of the United States in the year that Citibank was founded?

Recursion Round 1 (in latent space):
Planner → ***Critic*** → ***Solver*** →

Recursion Round 2 (in latent space):
Planner → ***Critic*** → ***Solver*** →

Recursion Round 3 (with final output):
Planner → ***Critic*** → ***Solver*:**
To answer this question, I need to find out the year Citibank was founded and then identify the President of the United States in that year.
First, I will search for the founding year of Citibank.

<search>When was Citibank founded?</search>
<result>
[1] Citigroup - Wikipedia
Citibank was chartered by the State of New York on June 16, 1812, with $2 million of capital. · · ·
[2] What is Brief History of Citi Company? - Matrix BCG
The genesis of Citi, originally named the City Bank of New York, dates back to June 16, 1812, when it received its charter in New York City. The bank commenced operations on September 14, 1812. · · ·
[3] Citibank - Students | Britannica Kids | Homework Help
Second largest bank in U.S., with about 270 branch banks in New York City area. Present name since 1976. · · ·
[4] A bank with history and tradition - Citi.com
Citibank opened for business in 1812 in New York under its original name City Bank of New York. Since then, Citi has been a major player in some of the world's most innovative breakthroughs in banking. · · ·
</result>

Based on the search results, Citibank (originally City Bank of New York) was founded in **1812**. Specifically, it was chartered on June 16, 1812, and opened for business on September 14, 1812. Now I need to find out who was the President of the United States in 1812.

<search>President of the United States 1812</search>
<result>
[1] The Election of 1812 | American Battlefield Trust
On October 14, 1812, incumbent president, James Madison, wrote to his friend and former president, · · ·
[2] The War of 1812: The Nation Braces for War
James Madison (1751-1836) James Madison, the Father of the Constitution and the Bill of Rights, was President of the United States during the War of 1812. · · ·
[3] Key American Personalities | 1812 Virtual Exhibition
As secretary of state under President Thomas Jefferson, future president James Madison had been deeply involved in the events leading to the war. · · ·
[4] James Madison - Wikipedia
Madison was elected president in 1808. Motivated by a desire to acquire land held by Britain, Spain, and Native Americans, Madison led the United States into the War of 1812. · · ·
</result>

The search results indicate that James Madison was the President of the United States in 1812. So, the answer is $\boxed{James\ Madison}$ *Correct Answer* ✓

H Case Study on RecursiveMAS across Different Recursion Rounds

Output of RecursiveMAS in Recursion Round 1 (On MATH500)

Question:
For how many positive integers $n > 1$ is it true that 2^{24} is a perfect n^{th} power?

Final Answer of RecursiveMAS:
First, let's express k in terms of its prime factorization. Since 2^{24} is a power of 2, k must also be a power of 2. Let $k = 2^m$ for some integer m. Then we have:

$$2^{24} = (2^m)^n = 2^{mn}$$

This equation tells us that $24 = mn$. We need to find all pairs of positive integers (m, n) such that $mn = 24$ and $n > 1$.
Let's list the factor pairs of 24:

$$(1, 24), (2, 12), (3, 8), (4, 6), (6, 4), (8, 3), (12, 2), (24, 1)$$

We discard the pairs where $n = 1$ because we are looking for $n > 1$. The remaining pairs are:

$$(2, 12), (3, 8), (4, 6), (6, 4), (8, 3), (12, 2)$$

Each pair (m, n) corresponds to a valid n greater than 1. Therefore, the possible values of n are 2, 3, 4, 6, 8, and 12.
Counting these values, we find there are 6 possible values for n.
Thus, the number of positive integers $n > 1$ for which 2^{24} is a perfect n-th power is $\boxed{6}$.
Incorrect Answer ✗

Output of RecursiveMAS in Recursion Round 2 (On MATH500)

Question:
For how many positive integers $n > 1$ is it true that 2^{24} is a perfect n^{th} power?

Final Answer of RecursiveMAS:
First, let's express 2^{24} in terms of its prime factorization: 2^{24}
For 2^{24} to be a perfect n^{th} power, n must be a divisor of 24. This is because if $2^{24} = k^n$, then k must be a power of 2, say $k = 2^m$. Therefore, we have:

$$2^{24} = (2^m)^n = 2^{mn}$$

Equating the exponents, we get:

$$24 = mn$$

We need to find all pairs (m, n) such that $mn = 24$ and $n > 1$. Let's list the divisors of 24:

$$1, 2, 3, 4, 6, 8, 12, 24$$

Since $n > 1$, we exclude 1 from the list of divisors. The remaining divisors are:

$$2, 3, 4, 6, 8, 12, 24$$

Thus, the number of positive integers $n > 1$ for which 2^{24} is a perfect n^{th} power is: $\boxed{7}$
Correct Answer ✓

Output of RecursiveMAS in Recursion Round 3 (On MATH500)

Question:
For how many positive integers $n > 1$ is it true that 2^{24} is a perfect n^{th} power?

Final Answer of RecursiveMAS:
First, let's express 2^{24} in terms of its prime factorization:
For 2^{24} to be a perfect n-th power, n must be a divisor of 24. This is because if $2^{24} = k^n$, then k must be of the form 2^m for some integer m, and thus we have $(2^m)^n = 2^{mn} = 2^{24}$, this implies:

$$mn = 24$$

We need to find all positive divisors of 24. The divisors of 24 are:

$$1, 2, 3, 4, 6, 8, 12, 24$$

Since we are looking for $n > 1$, we exclude 1 from the list. The remaining divisors are:

$$2, 3, 4, 6, 8, 12, 24$$

Thus, the number of positive integers $n > 1$ for which 2^{24} is a perfect n-th power is: $\boxed{7}$
Correct Answer ✓

Output of TextMAS (On MATH500)

Question:
For how many positive integers $n > 1$ is it true that 2^{24} is a perfect n^{th} power?

Output of Planner Agent:
Ok, I need to provide a step-by-step plan to solve the problem.

- Step 1: Determine the prime factorization of 2^{24}.
- Step 2: Recognize that 2^{24} can be expressed as a perfect n^{th} power if the exponent 24 is divisible by n.
- Step 3: Find all positive integers $n > 1$ such that 24 is divisible by n.
- Step 4: Count the number of such integers n.
- Step 5: Conclude that the number of positive integers $n > 1$ for which 2^{24} is a perfect n^{th} power is equal to the number of divisors of 24 that are greater than 1.
- Step 6: List the divisors of 24 greater than 1 and count them.
- Step 7: The final answer is the count of such divisors.

Output of Critic Agent:
Here is a detailed plan for the problem:

Step 1: Analyze the problem
We are asked to find how many positive integers $n > 1$ satisfy the condition that 2^{24} is a perfect n^{th} power.

Step 2: Recall the definition of a perfect n^{th} power
A number x is a perfect n^{th} power if there exists an integer k such that $x = k^n$.

Step 3: Express 2^{24} as a power of some base
We can write $2^{24} = (2^m)^n$, where m and n are positive integers.

Step 4: Find the possible values of n
For 2^{24} to be a perfect n^{th} power, n must be a divisor of 24 (since $2^{24} = (2^m)^n$ implies n divides 24).

Step 5: Count the number of valid divisors
Find the total number of positive divisors of 24, and then subtract 1 to account for the restriction that $n > 1$.

Output of Solver Agent:
To determine for how many positive integers $n > 1$ the number 2^{24} is a perfect n-th power, we need to find all the divisors of 24, since 2^{24} can be written as

$$(2^k)^n$$

, where $kn = 24$. First, we list all the divisors of 24. The divisors of 24 are:

$$1, 2, 3, 4, 6, 8, 12, 24$$

Each divisor d of 24 corresponds to a possible value of n such that 2^{24} is a perfect n-th power. Specifically, if d is a divisor of 24, then $n = \frac{24}{d}$ will be an integer and $2^{24} = (2^d)^{24/d}$.
Let's list the corresponding n values for each divisor d: - If $d = 1$, then $n = \frac{24}{1} = 24$ - If $d = 2$, then $n = \frac{24}{2} = 12$ - If $d = 3$, then $n = \frac{24}{3} = 8$ - If $d = 4$, then $n = \frac{24}{4} = 6$ - If $d = 6$, then $n = \frac{24}{6} = 4$ - If $d = 8$, then $n = \frac{24}{8} = 3$ - If $d = 12$, then $n = \frac{24}{12} = 2$ - If $d = 24$, then $n = \frac{24}{24} = 1$ (but since $n > 1$, we exclude this case)
Thus, the possible values of n are $2, 3, 4, 6, 8, 12, 24$. There are 7 such values.
Therefore, the number of positive integers $n > 1$ for which 2^{24} is a perfect n-th power is: $\boxed{7}$
Correct Answer ✓

I Prompt Template for RecursiveMAS

Prompt Template for Sequential-Style RecursiveMAS

System Prompt for All Agents:
You are a helpful assistant.

User Prompt for Planner Agent:
You are a planner agent in a recursive multi-agent system. Here is the latent information from previous round: {Latent Thought Embeddings}. Given the latent information, you should output a step-by-step plan to solve the question: {Question}

User Prompt for Critic Agent:
You are a critic agent in a recursive multi-agent system. Here is the latent information from previous agent: {Latent Thought Embeddings}. Given the latent information, you should critique the initial plan and output an improved plan to solve the question: {Question}

User Prompt for Solver Agent:
You are a solver agent in a recursive multi-agent system. Here is the latent information from previous agent: {Latent Thought Embeddings} Given the latent information, you should solve the question and provide the final answer: {Question}
Solve the question and put the final answer inside \boxed{}.

Prompt Template for Mixture-Style RecursiveMAS

System Prompt for All Agents:
You are a helpful assistant.

User Prompt for Math Specialist Agent:
You are a math specialist agent in a recursive multi-agent system. Here is the latent information from previous round: {Latent Thought Embeddings} Given the latent information, you should provide a domain-specific answer for the question: {Question}

User Prompt for Science Specialist Agent:
You are a science specialist agent in a recursive multi-agent system. Here is the latent information from previous round: {Latent Thought Embeddings} Given the latent information, you should provide a domain-specific answer for the question: {Question}

User Prompt for Code Specialist Agent:
You are a code specialist agent in a recursive multi-agent system. Here is the latent information from previous round: {Latent Thought Embeddings} Given the latent information, you should provide a domain-specific answer for the question: {Question}

User Prompt for Summarizer Agent:
You are a summarizer agent in a recursive multi-agent system. Here is the latent information from the math specialist: {Math Specialist Latent Thought Embeddings}. Here is the latent information from the code specialist: {Code Specialist Latent Thought Embeddings}. Here is the latent information from the science specialist: {Science Specialist Latent Thought Embeddings}. Given the latent information from all previous specialists, you should aggregate their reasoning and provide the final answer to the question: {Question}
Put the final answer inside \boxed{}.

Prompt Template for Distillation-Style RecursiveMAS

System Prompt for All Agents:
You are a helpful assistant.

User Prompt for Expert Agent:
You are an expert agent in a recursive multi-agent system. Here is the latent information from previous round: {Latent Thought Embeddings}. Given the latent information, you should provide expert-level guidance to solve the question: {Question}

User Prompt for Learner Agent:
You are a learner agent in a recursive multi-agent system. Here is the latent information from previous agent: {Latent Thought Embeddings}. Given the latent information, you should use the expert guidance to solve the question and provide the final answer: {Question}
Put the final answer inside \boxed{}.

Prompt Template for Deliberation-Style RecursiveMAS

System Prompt for All Agents (with Tool-Calling Instructions):
You are a helpful assistant that can solve the given question with the help of external tools, including a web search tool and a Python interpreter. For each question, you should reason carefully and use tools when they are necessary. You can invoke the web search tool to retrieve factual information, and use the Python interpreter to perform calculations or execute code. The search query and search result should be enclosed within `<search>` `</search>` and `<result>` `</result>` tags respectively. The Python code and execution result should be enclosed within `<python>` `</python>` and `<result>` `</result>` tags respectively.

User Prompt for Reflector Agent:
You are a reflector agent in a recursive multi-agent system. Here is the latent information from previous round: {Latent Thought Embeddings}. Given the latent information, you should reflect on the current solution and provide an improved solution to the question: {Question}

User Prompt for Tool-Caller Agent:
You are a tool-caller agent in a recursive multi-agent system. Here is the latent information from the Reflector agent: {Latent Thought Embeddings}. Given the latent information, you should solve the question: {Question}
Put the final answer inside \boxed{}.

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: Our claims are properly summarized in our Introduction section. All claims made match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.

   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: We properly discuss the Limitations in Appendix F.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: We provide detailed discussions in Appendix A

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: We provide detailed experiment setups to ensure reproducibility in Section 5 and Appendix B.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Code implementations are provided in Supplementary Materials for reference. We also provide well-written instruction files to reproduce our results, and discuss all experimental setups in detail in the Appendix.

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: We have provided detailed experiment settings in the Appendix B.

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: All experimental results are averaged over five independent runs, and we report the standard deviation of RecursiveMAS in Section 5.

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: We provide detailed information on experiments' compute resources in Appendix B.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have ensured to conform with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: Please refer to Appendix F. This paper presents foundational research in Agentic AI, and we do not see any direct use of our proposed method to make any positive or negative societal impact.

    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: This work presents no direct misuse risk.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We have properly cited all asset sources.

    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: This paper does not release new assets.

    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: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.