Latent Collaboration in Multi-Agent Systems

背景：

多智能体系统领域 —— 从孤立的、以模型为中心的推理，转向多个交互模型之间的协同任务。在这类基于 LLM 的多智能体系统中，**自然语言（或文本）**通常扮演着 "通用语" 的角色 —— 作为承载每个智能体内部思考、实现不同智能体间通信的共同媒介。

简单来讲就是一个问题：以前多个 AI 模型协作，都是靠文字交流 ，比如一个模型输出文字结论，另一个模型再读文字继续处理。但这样有两个麻烦：**一是文字能承载的信息有限（hiddenstates中蕴含了回答的置信度，困惑度等等信息），二是生成和解读文字特别费时间、费资源（编码解码也会占用大量时间）。**

这个论文就想尝试一个事

💡 *Can multi-agent systems achieve pure latent collaboration?*

LatentMAS 核心思路

1. 每个 Agent 思考时，不生成文字，而是**产生 "隐藏的想法"(利用hidden states)**，就是模型内部的连续数据（**最后一层隐藏状态**），这种 "想法" 能包含比文字更丰富的信息。
2. 模型之间靠 "**共享工作记忆**" 传递信息：**把各自的 "隐藏想法" 和输入内容存在特定缓存里**，后面的模型直接读取这个缓存，不用再解读文字，信息一点都不损耗。
3. "隐藏想法" 和模型能接收的输入格式不太匹配（**模型接受 都是embedding，如果存的都是last hiddenstates都会有分布上的不一致**），于是加了个简单的 "**转换工具**"，让 "想法" 能顺利被其他模型接收，还不用额外训练。

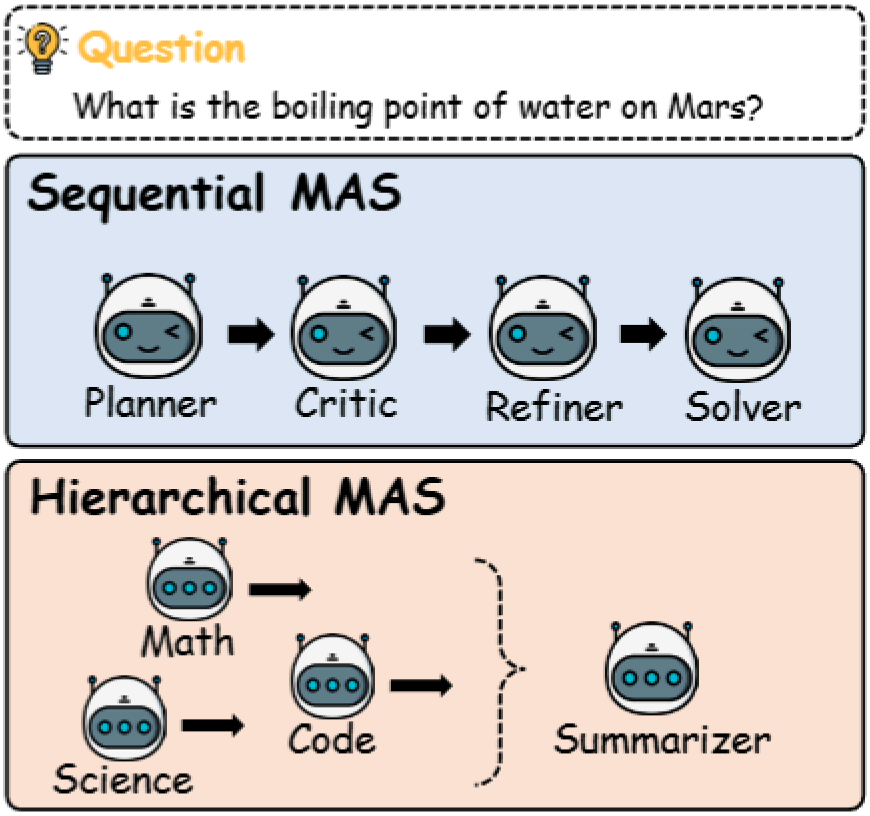

Figure 2 | Illustration of sequential and hierarchical MAS.

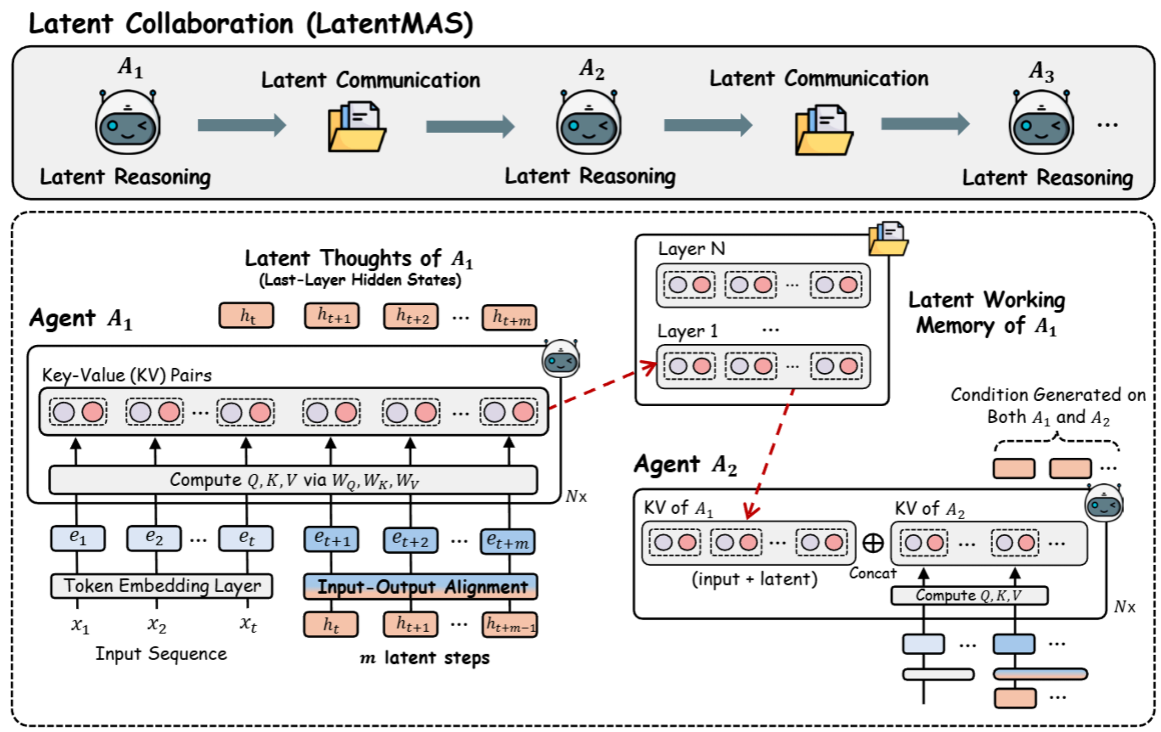

Figure 3 | **Overview of LatentMAS.** Each LLM agent in the system first generates latent thoughts through last-layer hidden states, then transfers information layer-wise via shared latent working memory stored in KV-caches, enabling completely system-wide latent collaboration.

其中 **sequential MAS** 采用了 “**智能体链条**” 的设计，由 4 个大型语言模型（LLM）智能体组成，分别是 planne、critic、refine和 solver。这 4 个智能体各司其职、互补配合，按顺序一步步推进工作（比如先规划、再点评、再优化、最后求解），形成一个流水线式的协作流程。在该架构中，每个智能体的思维链输出会与问题 q 一同作为下一个智能体的输入。在**hierarchical MAS**中，采用了领域专业化设计。多个大型语言模型智能体扮演不同领域专家的角色，每个智能体从自身学科视角出发，独立对问题 q 进行推理。随后，一个汇总智能体会接收所有中间响应及原始问题 q，通过分层聚合的方式合成并优化最终答案。

方法：

1、auto-regressive Latent Thoughts Generation in Agents.

每个agents: $E = [e_1, e_2, \ldots, e_t]$ 包含 problem，instruction，

回答 $H = [h_{t+1}, h_{t+2}, \ldots, h_{t+m}]$ 最后一层hiddenstates

Input-Output Distribution Alignment

由于新生成的表征序列 H 是密集的高层级表征，若直接将其作为输入嵌入插入浅层，可能会导致分布外激活。为在无需训练的前提下缓解该问题，提出一种线性对齐算子，**可将最后一层隐藏状态映射回有效的输入嵌入空间**。

$$e = hW_a, \quad \text{where } W_a \approx W_{\text{out}}^{-1} W_{\text{in}},$$

2、Working Memory Preservation and Thoughts Transfer across Agents.

$$\begin{aligned} \mathcal{M}_{A_1} &= \left\{ \left(K_{A_1,\text{cache}}^{(l)}, V_{A_1,\text{cache}}^{(l)} \right) \middle| \, l = 1, 2, \ldots, L \right\}, \\ &\text{where } K_{A_1,\text{cache}}^{(l)} = [K_{A_1,1}^{(l)}, \ldots, K_{A_1,t+m}^{(l)}], \quad V_{A_1,\text{cache}}^{(l)} = [V_{A_1,1}^{(l)}, \ldots, V_{A_1,t+m}^{(l)}]. \end{aligned} \tag{4}$$

A2 智能体对 A1 工作记忆的整合方式

接下来，后续智能体 **A2** 将整合来自智能体 **A1** 的工作记忆，记为 M_{A_1} 。

在 **A2 生成隐式思维（即最后一层隐藏状态）之前**，我们通过分层拼接的方式更新其键值（KV）缓存

结果

Table 1 | **Main results of LatentMAS on 6 general tasks under the Sequential MAS setting.** We report 3 metrics in total, including task accuracy (%, "**Acc.**"), total output token usage ("**Token**"), and end-to-end inference speed (time(s) / run, "**Speed**"). We compare LatentMAS with both TextMAS and single-model ("Single") baselines. For each metric, we **bold** the better performance and visualize LatentMAS gains over TextMAS in the **Improve** columns.

Tasks	Metrics	Qwen3-4B			Improve	Qwen3-8B			Improve	Qwen3-14B			Improve
		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS	
Sequential MAS Setting													
ARC-E	Acc.	95.4	96.4	**98.6**	↑ 2.2	95.6	99.1	98.8	↓ 0.3	97.2	99.0	**99.4**	↑ 0.4
	Token	724	2420	**581**	↓ 76.0%	656	2085	**490**	↓ 76.5%	608	1670	**224**	↓ 86.6%
	Speed	369	2874	**512**	×5.6	404	3702	**1759**	×2.1	551	9171	**2124**	×4.3
ARC-C	Acc.	89.2	90.0	**92.3**	↑ 2.3	91.0	94.6	94.4	↓ 0.2	92.6	95.9	95.6	↓ 0.3
	Token	913	2678	**718**	↓ 73.2%	846	2252	**529**	↓ 76.5%	773	2985	**426**	↓ 85.7%
	Speed	97	1579	**260**	×6.1	266	2059	**703**	×2.9	338	5125	**1136**	×4.5
GSM8K	Acc.	82.4	89.8	88.2	↓ 1.6	81.1	92.3	**93.8**	↑ 1.5	83.7	93.8	**95.2**	↑ 1.4
	Token	1136	3172	**607**	↓ 80.9%	1280	2324	**860**	↓ 63.0%	1118	3324	**644**	↓ 80.6%
	Speed	469	1970	**375**	×5.3	449	1739	**543**	×3.2	536	3729	**1952**	×1.9
MedQA	Acc.	47.7	65.3	**66.3**	↑ 1.0	53.0	75.0	**75.3**	↑ 0.3	64.7	80.3	**80.7**	↑ 0.4
	Token	2134	3962	**1685**	↓ 57.5%	2098	4260	**1555**	↓ 63.5%	1746	3444	**1841**	↓ 46.5%
	Speed	236	1267	**438**	×2.9	476	1923	**928**	×2.1	1360	4142	**1420**	×2.9
MBPP+	Acc.	63.5	69.8	**73.5**	↑ 3.7	64.8	69.5	**74.6**	↑ 5.1	68.5	72.8	**75.7**	↑ 2.9
	Token	1634	4420	**1339**	↓ 69.7%	2053	3695	**1164**	↓ 68.5%	1858	4971	**1621**	↓ 67.4%
	Speed	523	2148	**577**	×3.7	1064	3628	**1275**	×2.8	2410	8728	**2400**	×3.6
HumanEval+	Acc.	75.0	79.7	**79.9**	↑ 0.2	74.4	80.5	**80.5**	↑ 0.0	76.8	81.1	**86.5**	↑ 5.4
	Token	2380	5987	**1775**	↓ 70.4%	2507	4593	**1866**	↓ 59.4%	2366	5934	**2042**	↓ 65.6%
	Speed	274	1044	**350**	×3.0	502	1619	**497**	×3.3	1084	4062	**1285**	×3.2

Table 2 | **Main results of LatentMAS on 6 general tasks under the Hierarchical MAS setting.** We report accuracy, token usage, and end-to-end speed, and highlight the performance gains following the same evaluation protocol as in Table 1.

Tasks	Metrics	Qwen3-4B			Improve	Qwen3-8B			Improve	Qwen3-14B			Improve
		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS	
Hierarchical MAS Setting													
ARC-E	Acc.	95.4	97.1	96.8	↓ 0.3	95.6	98.2	**98.3**	↑ 0.1	97.2	98.3	98.7	↑ 0.4
	Token	724	2054	**363**	↓ 82.3%	656	2237	**308**	↓ 86.2%	608	2752	**619**	↓ 77.5%
	Speed	369	2239	**591**	×3.8	404	3619	**1779**	×2.0	551	7102	**1884**	×3.8
ARC-C	Acc.	89.2	92.5	91.7	↓ 0.8	91.0	93.3	**93.9**	↑ 0.6	92.6	95.3	**95.5**	↑ 0.2
	Token	913	2674	**447**	↓ 83.3%	846	2854	**344**	↓ 87.9%	773	2167	**295**	↓ 86.4%
	Speed	97	1275	**299**	×4.3	266	2034	**714**	×2.8	338	4283	**1090**	×3.9
GSM8K	Acc.	82.4	89.4	88.4	↓ 1.0	81.1	90.4	89.5	↓ 0.9	83.7	90.8	**91.6**	↑ 0.8
	Token	1136	3098	**555**	↓ 82.1%	1280	2370	**353**	↓ 85.1%	1118	3021	**495**	↓ 83.6%
	Speed	469	1878	**360**	×5.2	449	1365	**702**	×1.9	536	3675	**1631**	×2.3
MedQA	Acc.	47.7	65.0	**67.3**	↑ 2.3	53.0	76.3	77.0	↑ 0.7	64.7	78.0	78.3	↑ 0.3
	Token	2134	6702	**1015**	↓ 84.9%	2098	6893	**1007**	↓ 85.4%	1746	5473	**899**	↓ 83.6%
	Speed	236	1495	**557**	×2.7	476	3387	**964**	×3.5	1360	7591	**1250**	×6.1
MBPP+	Acc.	63.5	69.3	70.6	↑ 1.3	64.8	71.9	72.2	↑ 0.3	68.5	73.0	73.8	↑ 0.8
	Token	1634	6782	**1339**	↓ 80.3%	2053	7703	**1264**	↓ 83.6%	1858	7458	**1187**	↓ 84.1%
	Speed	523	1766	**489**	×3.6	1064	3898	**1387**	×2.8	2410	9162	**2507**	×3.7
HumanEval+	Acc.	75.0	76.2	**79.3**	↑ 3.1	74.4	76.8	**78.0**	↑ 1.2	76.8	84.1	**86.6**	↑ 2.5
	Token	2380	8127	**1373**	↓ 83.1%	2507	8768	**1274**	↓ 85.5%	2366	8114	**1512**	↓ 81.4%
	Speed	274	931	**333**	×2.8	502	1809	**439**	×4.1	1084	3988	**1188**	×3.4

Table 3 | **Main results of LatentMAS on 3 reasoning-intensive tasks under both Sequential and Hierarchical MAS settings.** We report accuracy, token usage, and end-to-end speed, and highlight the performance gains following the same evaluation protocol as in Table 1.

Tasks	Metrics	Qwen3-8B			Improve	Qwen3-14B			Improve
		Single	TextMAS	LatentMAS		Single	TextMAS	LatentMAS	
Sequential MAS Setting									
AIME24	Acc.	50.0	53.3	**56.7**	**↑ 3.4**	63.3	63.3	**66.7**	**↑ 3.4**
	Token	12891	38596	**8953**	**↓ 76.8%**	11263	32092	**10593**	**↓ 67.0%**
	Speed	421	2808	**688**	**×4.1**	1018	4554	**1149**	**×4.0**
AIME25	Acc.	46.7	53.3	**53.3**	↑ 0.0	56.7	60.0	**63.3**	**↑ 3.3**
	Token	14692	45088	**8699**	**↓ 80.7%**	11298	44618	**11402**	**↓ 74.4%**
	Speed	450	3150	**820**	**×3.8**	1040	5184	**1473**	**×3.5**
GPQA-Diamond	Acc.	39.9	43.4	**45.5**	**↑ 2.1**	48.5	51.5	**52.0**	**↑ 0.5**
	Token	6435	17986	**4571**	**↓ 74.6%**	5547	12676	**5454**	**↓ 57.0%**
	Speed	813	5771	**854**	**×6.8**	1043	9714	**1475**	**×6.6**
Hierarchical MAS Setting									
AIME24	Acc.	50.0	53.3	53.3	↑ 0.0	63.3	70.0	**73.3**	**↑ 3.3**
	Token	12891	42629	**7526**	**↓ 82.3%**	11263	29025	**10230**	**↓ 64.8%**
	Speed	421	3132	**776**	**×4.0**	1018	5718	**1089**	**×5.3**
AIME25	Acc.	46.7	50.0	50.0	↑ 0.0	56.7	66.7	66.7	↑ 0.0
	Token	14692	53929	**13230**	**↓ 75.5%**	11298	50003	**9527**	**↓ 80.9%**
	Speed	450	3488	**616**	**×5.7**	1040	6019	**1056**	**×5.7**
GPQA-Diamond	Acc.	39.9	43.0	**46.9**	**↑ 3.9**	48.5	52.0	**53.0**	**↑ 1.0**
	Token	6435	22450	**3395**	**↓ 84.9%**	5547	20931	**3606**	**↓ 82.8%**
	Speed	813	6108	**798**	**×7.7**	1043	9119	**1458**	**×6.3**

效率提升

隐式思维生成所需的隐式步骤数大幅减少，而逐令牌文本生成则需要远更多的解码步骤。

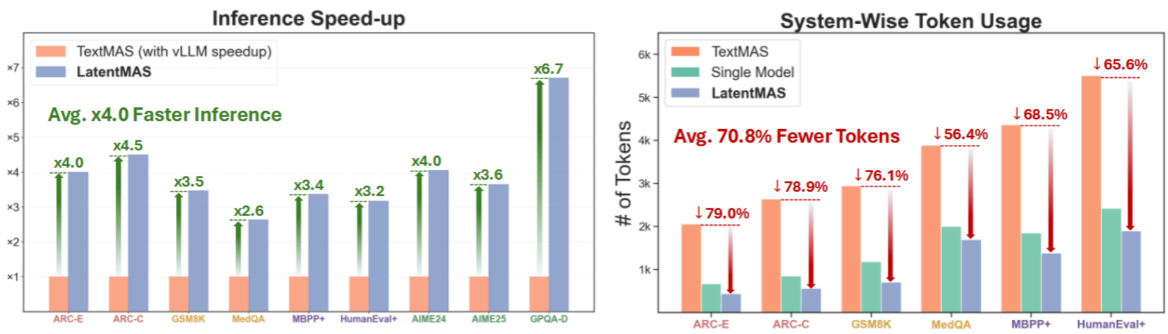

Figure 4 | Efficiency gains of LatentMAS over single model and TextMAS under the sequential MAS setting. **Left:** LatentMAS achieves substantially faster end-to-end inference, even though all baselines are accelerated with vLLM backend. **Right:** LatentMAS requires far fewer system-wise token usage.

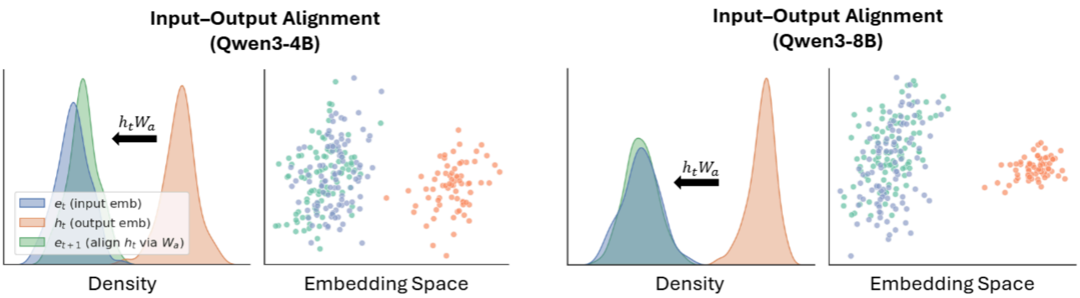

Figure 6 | **Effectiveness of the input-output alignment W_a on MedQA.** Unaligned output embeddings (h_t) drift away from the original input embeddings (e_t), while the aligned vectors (e_{t+1}) realign with e_t, demonstrating that W_a preserves embedding-space structure and prevents representation drift.

两项关键发现：

（i）LatentMAS 的最后一层嵌入与 TextMAS 的令牌嵌入在嵌入空间中几乎处于同一区域，这表明隐式思维编码的语义表征与正确文本响应的语义表征相似；

（ii）LatentMAS 的最后一层嵌入在很大程度上覆盖了 TextMAS 的令牌嵌入分布，这意味着隐式思维比离散令牌具有更丰富的多样性和更强的表达能力。

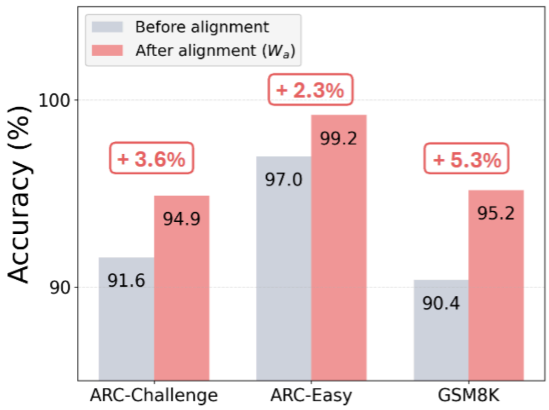

Figure 7 | Downstream performance before/after applying the input-output alignment W_a.

m	0	10	20	40	80	160
ARC-C	91.3	93.4	93.4	**94.9**	94.8	93.7
ARC-E	94.7	98.9	98.9	99.4	**99.6**	98.3
GSM8K	85.6	90.3	90.9	91.4	**92.0**	91.9

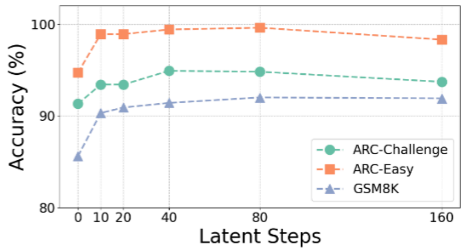

Figure 8 | Effectiveness of different latent step depths of LatentMAS on downstream performance.